# ANALYSIS AND COMPARATIVE STUDY OF VARIOUS SOFTWARE DEVELOPMENT PROCESS MODELS

**A THESIS IS SUBMITTED TO**
**SAURASHTRA UNIVERSITY, RAJKOT**
**FOR THE AWARD OF DOCTOR OF PHILOSOPHY IN**
**COMPUTER SCIENCE IN THE FACULTY OF SCIENCE**

**: SUBMITTED BY :**

**AMLANI RADHIKA DINESHKUMAR**
**DR.V.R.GODHANIYA IT COLLEGE, PORBANDAR**

**UNDER THE GUIDANCE OF**

**DR. KISHOR ATKOTIYA**
**HEAD,**
**DEPARTMENT OF COMPUTER SCIENCE,**
**J. H. BHALODIA WOMEN'S COLLEGE,**
**RAJKOT - 360 002**

**SEPTEMBER-2013**

# CERTIFICATE

I hereby certify that **Miss. Radhika Dineshkumar Amlani** has completed his thesis for doctorate degree entitled **"ANALYSIS AND COMPARATIVE STUDY OF VARIOUS SOFTWARE DEVELOPMENT PROCESS MODELS"**. I further certify that the research work done by him is of his own and original and is carried out under my guidance and supervision. For the thesis that he is submitting, he has not been conferred any degree, diploma or distinction by either the Saurashtra University or any other University according to best of my knowledge.

Place: Rajkot
Date:

**Dr. Kishor Atkotiya**
**Head,**
**Department of Computer Science,**
**J. H. Bhalodia Women's College,**
**Rajkot - 360 002**
**INDIA**

# CERTIFICATE

Myself, Radhika Dineshkumar Amlani, a student Ph.D. at Saurashtra University, Rajkot, certify that the developed model: LL Model (which is a short name of Lessons Learned Model), described in this research study is based on the study of SDLC waterfall model, v-shaped model, spiral model, RUP model, RAD model, JAD model, Scrum model, XP model, prototype model and iterative and incremental model and comparative study by analysis and described in the thesis. It is also based on the literature survey, bibliographical references and study of the web sites in respect of related areas.

Apart from these, all the analysis, hypothesis, inferences and interpretation of development strategies have been my own and original creation. The model has been prototyped to a domain, which is my own and original creation. Moreover, I declare that the work done in the thesis, either the Saurashtra University or any other university has not conferred any degree, diploma or distinction on me before.

Place: Rajkot
Date:

**Amlani Radhika Dineshkumar**

# ACKNOWLEDGEMENT

I express my profound sense of gratitude to **Dr. K.H. Atkotiya** my research guides, who provides me undeviating encouragement, indefatigable guidance and valuable suggestions throughout the research study.

I take opportunity to express my deep sense of gratitude to **Dr. M. K. Padalia**, Vice-Chancellor of the Saurashtra University for his consistent encouragement to the research and development.

I express my deepest gratitude to the **Dr. G. C. Bhimani,** Associate Professor, Department of Business Management, Saurashtra University for his profound encouragement for my research work.

I am thankful to **Mr. Nilesh Joshi,** Project Manager, MetrixOne System Pvt. Ltd., who has implemented my concept and incorporate with their system and give me result of the implementation.

I am thankful to **Ms. Jaldeepa Joshi,** senior information developer at Symantec, and **Mr. Jay Palan,** system engineer at Infosys. They provided me valuable information and insight into various important issues related to the research study.

I also give my sincere thanks to the Department of Computer Science, Saurashtra University, to provide me platform for the practical study of my research work. I am also thankful to the administrative staff of the department, who has always been a support of inspiration during my entire work.

I am so thankful to Virambhai Godhaniya, chairman of Dr. V. R. Godhaniya IT College, Bharatbhai Visana, Managing Trusty of Dr. V. R. Godhaniya IT College and all the trusties. I am also thankful to all the staff of my college who have been supported me always.

I am deeply thankful to the reviewer of my thesis. I am highly indebted to my parents, my husband, brothers, sisters and all my relatives and friends who constantly inspired me.

Rajkot
Amlani Radhika Dineshkumar

# ABSTRACT

With this research work, researcher has tried to give a new enhanced model based on component based SDLC model. There are so many software development process models available in the market to develop the software. It is but natural that the development of any software is quite lengthy process. There are so many up and downs are to be faced during the software development life cycle process.

So, researcher has tried to suggest a model which introduces a concept of “lessons learned”. This concept is a process of learning from the now running or already completed previous projects. Such the learning could be good experience or bad experience observed, faced or solved during the development process. This would ease the development process. This would lead to problem-free development which could save the time of development and management team. System could be developed at decided time and deliver to the customer.

It is always nice, if we learn from our mistakes, but it would be intelligence, if we could learn from the others mistakes. And in actual, the expertise means the learning from the mistakes as well as remembering and reutilizing the better matter we have found during experiences of our own or the others. So, for the smooth development of the system, learning from the past experience would lead to the quick and sure success. So, that researcher has suggested a model named “LLMODEL”.

# LIST OF TABLES

# LIST OF FIGURES

# TABLE OF CONTECT

# 1. Introduction to the Software Development

Software plays a vital role in utilizing a computer or a computerized device. It delivers different tasks as well as allows us to use, share and maintain the resources. As a product, it delivers the computing potential embodied by computer hardware. Software is an information transformer like producing, managing, modifying, acquiring, or transmitting information when it resides within a cellular phone or operates inside a computer like mainframe computer. Software acts as a foundation for the control of the computer, the communication of information and the creation and control of other program. At the other end we may use it to produce a product i.e. Programming Software, Maintenance Tools, Patches, Mechanical Robotics or a simple automation of them etc. for all these a human being needs to develop required software.

Software development process requires a skill in the field of computer as well as in the field for which the software is to be developed. No doubt the knowledge of the field you may gather later but the knowledge of Software Development Processes must be clear at every stage. After several experiences and by research in the field of software development few Software Development Process Models are derived. And are enough capable to process the software development processes.

But, in nearby future there may be vast improvement in hardware performance, profound changes in computing architectures, and vast expansion of memory and storage capacity, and a wide variety of exotic input and output options. These all precipitated more sophisticated and complex computer based system. So there are more things to be done incorporate with software product.

Because, the Software development process has historically a labor and time intensive task, either done through complete experimental studies or in an automated fashion using techniques such as logging and analysis of command shell operations. While experimental studies have been fruitful,

data collection has proven to be tedious and time consuming. Existing automated approaches have very detailed, low level but not rich results. We are interested in process discovery in large, globally, sometimes across several stages of the software lifecycle in parallel this presents a challenge.

For these some related work is to be done in the term of Problem Specification, Process Discovery and Modeling, Process Re-enactment for Deployment, Validation, and Improvement etc. The approach to discovery, modeling, and re-enactment depends on a variety of informal and formal process representations.

As informal process descriptions, flow graphs as informal but semi structured process representations which we transformed into a formal C. Jensen, W. Scacchi process representation language guided by a process meta-model and support tools. These informal representations together with a process meta-model then provide a basis for constructing formal process descriptions. Thus demonstration of a more automated process discovery, modeling, and indulging environment studies that integrates these capabilities and mechanisms into a smoother and more automated environment is the next step in this research. We anticipate that such an environment will yield additional values for tool assistance in process data collection and analysis.

## 1.1 INTRODUCTION TO SOFTWARE

Software is the general term for information that's recorded onto some kind of medium. For example, when you go to the video shop and rent or buy a tape or DVD, what you're actually getting is the software that's stored on that tape or disk. VCR or DVD player are hardware devices that are used to read software from a tape or disk and projecting it onto your TV screen, in the form of a movie.

Your computer is a hardware device that reads software too. Most of the software on your computer comes in the form of programs. A program consists of "instructions" that tell the computer what to do, how to work. Just as there are thousands of albums you can buy on CD for your stereo, and

thousands of movies you can buy to play on your VCR or DVD player, there are thousands of programs that you can buy to run on your computer.

When you purchase a computer, you don't automatically get every program produced by every software company in the world. You usually get some programs. For example, when you buy a computer it will probably have an operating system (like Windows XP, Windows 7) already installed on it.

If you do purchase a specific program, it would be to achieve some specific task. For example, you might use a graphics program to touch up photos, or you might use a word processing program to write text. You're using your Web browser program to find any information on internet. Just as there are so many different brands of toothpaste, there are so many different brands of word processing programs, graphics programs, and Web browsers.

Software is a common term for the different kinds of programs used to operate computers and related devices. Software can be thought of as the variable part of a computer and hardware the invariable part. Computer instructions or data, anything that can be stored electronically is software. The storage devices and display devices are hardware.

The difference between software and hardware is sometimes confusing because they are so integrally linked. Clearly, when you purchase a program, you are buying software. But to buy the software, you need to buy the disk (hardware) on which the software is recorded.

Computer software, or just software, is a group of computer programs and related data that provide the instructions for telling a computer what to do and how to do it. In other words, software is a theoretical entity which is a set of computer programs, procedures, and associated documentation concerned with the operation of a data processing system. We can also say software refers to one or more computer programs and data held in the storage space of the computer for some purposes. In other words, software is a set of programs, procedures, algorithms and its documentation. Program software serves the purpose of the program it implements, either by directly providing instructions to the computer hardware or by serving as input to

another part of software. The term was coined to contrast to the old term hardware (meaning physical devices). In contrast to hardware, software is intangible, meaning it "cannot be touched". Software is also sometimes used in a more narrow sense, meaning application software only. Sometimes the term includes data that has not traditionally been associated with computers, such as film, tapes, and records.

Software includes all the various forms and roles that digitally stored data may have and play in a computer (or similar system), regardless of whether the data is used as code for a CPU, or other interpreter, or whether it represents other kinds of information. Software covers a broad array of products that may be developed using different techniques such as ordinary programming languages, scripting languages, microcode.

Computer software is so called to differentiate it from computer hardware, which includes the physical interconnections and devices, needed to store and run the software. At the core level, executable code consists of machine language instructions specific to an individual processor. A machine language made of groups of binary values signifying processor instructions that change the state of the computer from its preceding state. Programs are an ordered sequence of instructions for changing the state of the computer in a particular sequence. It is usually written in high-level programming languages that are easier and more efficient for humans to use than machine language. High-level languages are compiled or interpreted into machine language object code. Software may also be written in an assembly language, essentially, a mnemonic representation of a machine language using a natural language alphabet. Assembly language must be assembled into object code via an assembler.

The types of software include web pages developed in languages and frameworks like HTML, PHP, Perl, JSP, ASP.NET, XML, and desktop applications like OpenOffice.org, Microsoft Word developed in languages like C, C++, Java, C#, or Smalltalk. Application software usually runs on underlying software operating systems such as Linux or Microsoft Windows. Software (or firmware) is also used in video games and for the configurable

parts of the logic systems of automobiles, televisions, and other consumer electronics.

### 1.1.1 Types of software

1) Application software is programs that include end-user applications of computers such as word processors or games, and ERP software for groups of users.
2) System software which includes operating systems and any program that supports application software. System software includes operating systems, which govern computing resources. Today large applications running on remote machines such as Websites are considered to be system software, because the end-user interface is generally through a graphical user interface, such as a web browser.
3) Middleware is sometimes used to explain programming that mediates between application and system software or between two different kinds of application software (for example, sending a remote work request from an application in a computer that has one kind of operating system to an application in a computer with a different operating system).
4) Testware is any software for testing hardware or a software package.
5) Firmware is low-level software often stored on electrically programmable memory devices. Firmware is given its name because it is treated like hardware and run by other software programs. Firmware often is not accessible for change by other entities but the developers' enterprises.

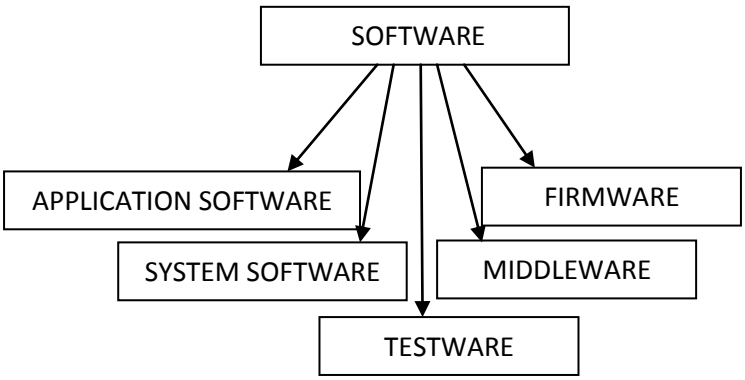



**FIGURE - 1 TYPES OF SOFTWARE**

## 1.2 HISTORY OF SOFTWARE[1]

The first theory about software was proposed by Alan Turing in his 1935 essay Computable numbers with an application to the Decision problem. The term "software" was first used in print by John W. Turkey in 1958. Colloquially, the term is often used to mean application software. In computer science and software engineering, software is all information processed by computer system, programs and data. The academic fields studying software are computer science and software engineering.

The history of computer software is most often traced back to the first software bug in 1946. As more and more programs enter the realm of firmware, and the hardware itself becomes smaller, cheaper and faster as predicted by Moore's law, elements of computing first considered being software, joining the ranks of hardware. Most hardware companies today have more software programmers on the payroll than hardware designers, since software tools have automated many tasks of Printed circuit board engineers. Just like the Auto industry, the Software industry has grown from a few visionaries operating out of their garage with prototypes. Steve Jobs and Bill Gates were the Henry Ford and Louis Chevrolet of their times, who capitalized on ideas already commonly known before they started in the business. In the case of Software development, this moment is generally

[1] http://en.wikipedia.org/wiki/Software

agreed to be the publication in the 1980s of the specifications for the IBM Personal Computer published by IBM employee Philip Don Estridge. Today his move would be seen as a type of crowd-sourcing.

Microsoft and Apple were able to thus cash in on 'soft' products. It is hard to imagine today that people once felt that software was worthless without a machine. There are many successful companies today that sell only software products, though there are still many common software licensing problems due to the complexity of designs and poor documentation, leading to patent trolls.

With open software specifications and the possibility of software licensing, new opportunities arose for software tools that then became the defacto standard, such as DOS for operating systems, but also various proprietary word processing and spreadsheet programs. In a similar growth pattern, proprietary development methods became standard Software development methodology.

## 1.3 SOFTWARE DEVELOPMENT

Software development[2] is also known as application development, software design, designing software, software application development, enterprise application development, or platform development. Software development is the development of a software product. The term "software development" may be used to refer to the activity of computer programming, which is the process of writing and maintaining the source code, but in a broader sense of the term it includes all that is involved between the conception of the desired software through to the final appearance of the software, ideally in a planned and structured process. Therefore, software development may include research, new development, prototyping, modification, reuse, re-engineering, maintenance, or any other activities that result in software products.

[2] http://en.wikipedia.org/wiki/Software_development

Software can be developed for so many of purposes, the three most common being to meet specific needs of a specific client/business (the case with custom software), to meet a supposed need of some set of potential users (the case with commercial and open source software), or for personal use (e.g. a software to automate a ordinary task). Embedded software development, that is, the development of embedded software such as used for controlling consumer products, requires the development process to be integrated with the development of the controlled physical product.

There are several different approaches to software development, much like the various views of political parties toward governing a country. Some take a more structured, engineering-based approach to developing business solutions, whereas others may take a more incremental approach, where software evolves as it is developed piece-by-piece. Most methodologies share some combination of the following stages of software development:

- Market research
- Gathering requirements for the proposed business solution
- Analyzing the problem
- Devising a plan or design for the software-based solution
- Coding of the software
- Testing the software
- Deployment
- Maintenance and bug fixing

All these stages are collectively referred as the software development lifecycle, or SDLC. Different approaches to software development may carry out these stages in different orders, or devote more or less time to different stages. The level of detail of the documentation produced at each stage of software development may also different. These all stages may also be carried out in turn, or they may be repeated over various cycles or iterations. The more extreme approach usually involves less time spent on planning and

documentation, and more time spent on coding and development of automated tests. More “extreme” approaches also encourage continuous testing throughout the whole development lifecycle, as well as having a working and bug-free product at all times.

There are considerable advantages and disadvantages to the different methodologies, and the best approach to solve a problem using software will often depend on the type of problem. A software development process is a format imposed on the development of a software product. Other names include software life cycle and software process. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process.

## 1.4 SOFTWARE DEVELOPMENT LIFE CYCLE

Life cycle means there is something that is created and it is ended at some saturation point, as after that created thing is of no use. So, it must meet to an end.

Software development life cycle is a life cycle of software that is created, used and then updated and if it is of no use then it is stopped using by anybody. It is a process or providing a framework through which software can be developed. It is a systematic approach to achieve desired result in terms of software development to meet up the desired requirement of the user.

Software developers of worldwide would agree that to structure a software takes more than just writing complex codes and implementing them in an environment. Developers usually begin their career in programming by developing programs or software according to their own plan and hope that someone would understand it. But once the developer is connected with a business or another software company, the creativity is restricted to business and consumer needs. The stress in creating accurate and efficient software is even more in the entrepreneurial stage.

To ensure this developers have come up with the right software for the specific need. Programmers have created steps on how a program could

specifically create. This will make sure everything is built according to plan and tested extensively before it could be deployed for public or formally utilize in business.

Under these state of affairs the term “Systems Development Life Cycle” was born. The need to create accurate and efficient software has led to the formalization on certain stages and phases on how a program should be built.

Simply put, SDLC or Systems Development Life Cycle is a series of steps observed by developers on building specific software. Developers follow some steps to ensure that they have the right software for the specific demand.

The history of the term “Systems Development Life Cycle” is not very clear but it naturally came into being since the 1960s when developers started to create programs specific to a certain need. Slowly, the term has been observed by different software development companies. From a simple structure of planning, building, testing and implementing, software companies have developed their own version of developing specific products for their clients. Each version of software development is called “Model”. As of this, there are number of known software development models observed by different software development companies. These development models could be applied in a specific situation to ensure the product created is a success, since not all models could be used in a certain application. Skills and experience in software development will also describe which model will work for software development.

SDLC (Software Development Life Cycle)[3] is the process of developing software through business needs, analysis, design, implementation and maintenance.

Software has to go through various phases before it is born which are as follows:

(i) Need of the software – A need of software may arise from the users to ease work. For example, any departmental store may need software to sell its newly arrived items. The owner of the company feels that he needs software that would help him in tracking his expenses and income as well as enhance

[3] http://istqbexamcertification.com/what-are-the-software-development-life-cycle-phases/

the selling process. This is how the concept is generated. The owner will exclusively tell the software company what type of software he would need. In other words, he will specify his requirements to the software company.

(ii) Study of Requirements – After the owner or user knows his requirements, then it is given to a software team or a company who will analyze the requirement and prepare requirement document that will describe each functionality that are required by the owner. The requirement document will be the core document for developers, testers and database administrators. In other words, this is the main document that will be referred by everyone. After the requirement documents, other detailed documents may be needed. For example, the architectural design which is a blueprint for the design with the necessary specifications for the hardware, software, people and data resources.

(iii) Development: After the detailed requirement documents, the developers start writing their code for their modules. On the other hand, the testers in the QA (Quality Assurance) Department start writing Test Plans (one module=1 test plan), test cases and get ready for testing.

(iv) Testing: Once the codes (programs) are generated, they are compiled together and to make application. This application is now tested by the software testers (QA Testers).

(v) Implement: After testing, the application goes to be deployed means; it will be handed over to the owner.

(vi) End: At some day, the owner will have say bye to the software either because the business grows and this software does not meet the demand or for some reason, the he does not need the software. That’s the end of it.

Above described are the steps which need to be followed to develop efficient and active software. Below given steps are the generalized phases of the software development.

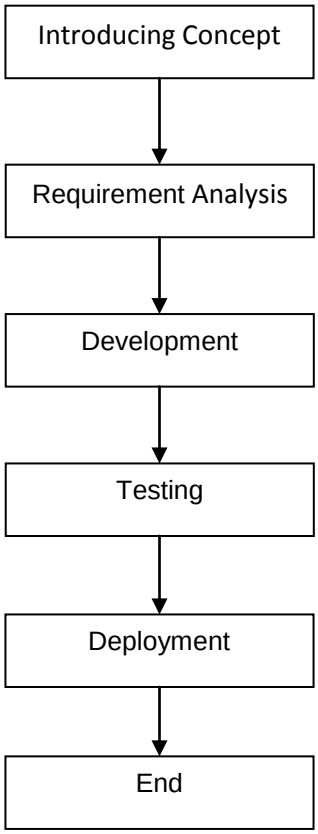



**FIGURE-2 GENERALIZED PHASES OF SOFTWARE DEVELOPMENT**

There are so many processes available in the market which is known as “Software Development Model”. All the processes have their own phases of the software development according to which to the software is developed. So, it is better to develop software by using suitable software development life cycle model.

## 1.5 NEED OF SOFTWARE DEVELOPMENT LIFE CYCLE

Software development life cycle describes the organization's standards through the creation and management of application.

Software can be very difficult and complex. We need the SDLC as a framework to guide the development to make it more systematic and efficient.

By using software development life cycle, developer can easily estimate about time and cost, taken by software development. SDLC provides easier time debugging and makes development process smoother.

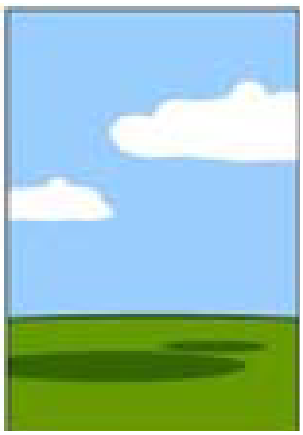
If software development is not based on any of the software development life cycle models then the scenario would be like following. Below given pictures show the importance of the software development life cycle.

Requirement Specification

Project Management

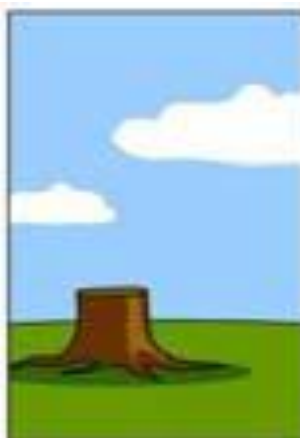
Requirement Analysis

Design & Development



Documentation

Installation



Maintenance

Developed Software

**FIGURE-3 SOFTWARE DEVELOPMENT WITHOUT ANY SDLC MODEL**

Above given pictures depict that if software development life cycle is not used to build software then developed software is not exactly as required. If software development life cycle is used to build software which result in a software and that software might meet the customer requirement to most extent.

Below given example[4] clearly shows the need of the software development life cycle.

When you pay your telephone bill your payment is processed by a system. That system has evolved over so many years and continues to evolve in order to meet the changing needs of the business. When the phone company cashes your cheque that cheque is also processed by a system which itself is evolving. These two systems are composed of manual activities and automated components. They also exist in the context of many other systems with which they must interface.

Each system works so well independently because it is composed of a correct set of tasks which result in well-defined outputs. Regardless of who is doing the task, the result is fundamentally the same.

Each system can interface with the other because the division of activities between the bank and the phone company are well defined, as are the interfaces. Thus, from which bank the cheque is drawn on that is no matter,

[4] http://www.benderrbt.com/Bender-SDLC.pdf

the process is the same; which Phone Company sends in the cheque is no matter, the process is the same.

The correctness and completeness of the task lists, the data, the division of responsibilities and the interface definitions are required because of the complexity of these systems. But, what is about the process that creates and maintains these systems?

Software systems development is, from a historical perspective, a very young profession. The first official programmer is probably Grace Hopper, working for the Navy in the mid-1940s. More realistically, commercial applications development did not really take off until the early 1960s. These initial efforts are marked by a craftsman-like approach based on what intuitively felt right. Unfortunately, too many programmers had poor intuition.

By the late 1960s it had become clear that a more disciplined approach was required. The software engineering techniques started coming into being. This finally brings us to the SDLC.

The process that developed from these early activities in improving is an understanding of the scope and complexity of the total development process. It became understandable that the process of creating systems required a system to do systems. This is the SDLC. That is the system that used to build and maintain software systems.

As with the phone billing system, an SDLC system is required as the development process is made of many complex tasks which must be done in the right order to produce a successful result. If there is no SDLC, each team must reinvent it based on its own experiences and judgments.

The difficulty of the tasks has led to increase in specialization. These specialists like data base analysts, network designers, testers, developers must have well-bounded tasks with well defined outputs and well-defined interfaces to the rest of the development team. This is provided by the SDLC.

Systems also have a long life. The billing system and cheque processing system will probably never be de-automated. These systems will live longer than their development teams. As staff turnover occurs, continuity is required in how the systems are supported. This is provided by the SDLC.

Another issue is systems integration. Can you imagine if the architects designing a skyscraper each had their own ways of creating and documenting the building's plans? The plumbing, wiring, heating systems, floors, walls, etc., would never come together. Most computer systems interface with other computer systems. The degree of system integration is rapidly increasing. The descriptions of functions and data at the interfaces must be produced and documented in a consistent manner; in much the same way as architects must have a standard way of documenting plans. This is provided by the SDLC.

One would not dream to process phone bills or cheques without a system due to the complexity of the process, which needs for specialization, which needs for continuity and needs to interface to the rest of the world. As we have gone through that the process of developing systems has the same characteristics. Therefore, we need a system to build computer systems - the SDLC.

It looks simple to build software based on a business need. For example, the business needs simple computing software embedded in their website. That is a simple problem that could be done by programmer, anytime. It is just a matter of using code to efficiently implement the software in a website.

But if one takes a look at it deeply, there are steps that should be followed before one can actually create the software. First one has to know what type of computing software and the components that should be added. Then they have to plan for the actual codes that will be used and test it extensively before implementation. Without any of these steps, something will definitely go wrong once the program is implemented.

## 1.6 IMPORTANCE OF SOFTWARE DEVELOPMENT LIFE CYCLE

With SDLC, software development companies or in-house developers will ensure the software released will have the following behaviours:

### 1.6.1 High quality software is created.

Building software is different from building well built software. Anyone could create software as there are available tools that do not even require deep knowledge in any of programming language. But with SDLC, Developers have to study deeply than doing work in shortcuts to create a specific tool that the customer wants.

### 1.6.2 Easy to implement project and control it

The task of developers does not finish when the software is deployed. Even though specific software is highly efficient when implemented, anything found wrong or a bug in the system should be worked on especially when the software did not go through beta testing. In this matter, SDLC will ensure that controls of the software are stable. This is usually done by creating better documentation to guide the developers in controlling the specific function of software.

### 1.6.3. Answer the need of the users

Software that is created should fulfil the exact needs of their clients. Having highly stable software is of no use if the intended users cannot use the software. SDLC will make sure the needs are fulfilled and could even provide more than customer required. The steps they will be using are leading towards creating software that is highly efficient as well as problem solving for better time management.

# 2. Current Status of SDLC Models

## 2.1 SDLC MODEL

System Development Life Cycle Model is used as a process of creating and altering current existing system. SDLC used in information system, systems engineering, and software engineering. SDLC can be thought of as a concept that used by many software development methodologies, which are currently available in market or software industry. SDLC gives a framework to create, plan and control any information system to be developed.

A Software Development Life Cycle Model is a set of activities together with an ordering relationship between activities performed in a manner that satisfies the ordering relationship that will produce desired product. SDLC Model is an abstract representation of a development process. In a software development effort the goal is to produce high quality software. The development process is, therefore, the sequence of activities that will produce such software. A software development life cycle model is broken down into distinct activities and specifies how these activities are organized in the entire software development effort[5].

SDLC explains a method, which is used by software engineers and software developers to build and implement all features or characteristic of information system. This features or characteristic describes requirements, validation, training and emphasizing ownership of the system. **Whenever SDLC is used, the main objective is to create a quality system that meets the primary need of the owner within defined timeline and cost constraints.** It also contains post installation stages like, deployment and maintenance, which has features like ease of use, installation of the software, for minimizing error.

There are so many SDLC based, software engineering models available in market now-a-days. Depending upon the suitability, the software engineering model can be used to put forward any software project. Each of the methodologies or models has different level of risk and benefits to manage

[5] http://www.ijcst.com/vol24/3/sanjana.pdf

with the project requirements, budget and estimated completion timeline. There are some models which are appropriate for large project, where some of them focus on lightweight process that allow rapid changes throughout whole software development life cycle.

The main objective of the SDLC would be to develop software which is exactly fit to the customer's requirement and working efficiently, within decided time period and this development would be cost effective.

## 2.2 ELABORATED PHASES OF SDLC[6]

The SDLC framework made of a multiple consecutive phases or steps that are to be followed in sequence by software developers and system designers. In each phase of System Development Life Cycle, all the phases are depend on the result of previous phase. The output of the previous phase becomes the input of next phase. The titles of the phases may be varying, depending on the development environment that include planning, analysis, and implementation.

### 2.2.1 System conceptualization

In this phase, the System's concept is developed. The thought of a system to be developed, is come into existence. Which kind of system is required, decides in this phase. A System is come into existence virtually. The idea of a system is born in this phase. This is base phase behind any new system to be developed, because the requirement of a system gives the idea of a new system to come into existence.

In this phase project's objective is to be determined whether to create new project or altering the current project. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals.

[6] http://www.ctg.albany.edu/publications/reports/survey_of_sysdev/survey_of_sysdev.pdf

The minimum information for each objective consists of a title and textual description, although additional information and references to external documents may be included.

In this phase all the planning should be done like what to do? What kind of features should be there in software? A kind of prototype would be developed at the planning stage. One more thing is taken into consideration like; the project is feasible in all aspect of the current situation. As a result long term project is planned.

The outputs of this phase are these, SDLC description, all the Software Project Management Plan (SPMP) and the associated project or component schedule. The schedule includes a detailed listing of activities for the upcoming requirement stage and high-level estimates of effort for the out stages.

### 2.2.2 System requirements and benefits analysis

This phase takes as its input the objectives identified in the above phase or system conceptualization. Each objective is refined into set of requirements.

System requirements are defined in this phase. Requirement of the System is actually there or not, is identify in this phase. Characteristics of System are decided in this phase. All the requirement or need should fulfill by new system. All the requirements to develop a system are analyze this phase. Weather the new system is beneficial over current system or not. All the pros and cons of the new system are analyzed in this phase. If the benefits are there of new system is much than current system then the idea to develop new system put forward.

In this phase all the requirement information are to be gathered. Which of them are taken into consideration is decided in this phase. Basis on this information, well-defined functions are created from the defined project goal.

This phase is concerned with establishing what the ideal system has to perform. However it does not determine how the software will be designed or built. Usually, the users are interviewed and a document called the user requirements document is generated.

The user requirements document will contain the system's functional, interface, performance, data, security, etc requirements as expected by the user. It is used by business analysts to communicate their understanding of the system to the users. The users carefully go through this document as this document would serve the purpose of guiding the system designers in the system design phase. The user acceptance tests are designed in this phase.

There are different methods for gathering requirements of both soft and hard methodologies including; interviews, questionnaires, document analysis, observation, use cases and status and dynamic views with users.

These requirements define the major functionality of the supposed application and defined the initial data entities. Major functions include critical processes to be managed for i.e. critical inputs, outputs and reports.

Each of these definitions is known as a Requirement. Requirements are identified by unique requirements identifiers and at minimum, contain a requirement title and textual description.

These requirements are fully described in the primary deliverables for this stage: the Software Requirements Document (SRD) and the Logical Database Description (LDD). The SRD contains complete descriptions of each requirement, including references to external documents, such as Use Cases. The LDD describes the major data entities of the project or component, along with their relationships to other entities and their user base.

The identifier associated with each requirement is also placed into the first version of the Requirements Traceability Matrix (RTM), along with the identifier of each goal from the parent project plan or component iteration plan.

### 2.2.3 Project adoption and project scoping

System requirements outputs will be inputs of this phase.

After thinking on all the pros and cons of the system requirements, the project is put into developing stage. All the requirements are analyze and decide to develop that project. Weather the current technology is enough to adopt a

new project. Which is new technology to adopt to cope with new project? What is the future scope of the project? All these things must be kept in mind.

Feasibility study of the system is done over here. There are four types of feasibility study. Like Technical feasibility, economical feasibility, operational feasibility, and organizational feasibility. Whether the system is feasible or not depending upon the respected requirements, it is decided.

### 2.2.4 System design

After getting positive result from previous phase the System is designed in detail. Outputs of the system requirements will be inputs of this phase.

Project is designed in this phase. Whole project/system design is developed. All operations and features are described in detail, which include technical specification. UML[7] is used in this phase when required. Process diagrams are prepared to facilitate the SDLC process with required documentation. In short, blue print of the whole system is developed in this phase. How the system will work, is decided.

System design is the phase where system engineers analyze and understand the business of the proposed system by studying the user requirements document. They plan our possibilities and techniques by which the user requirements can be implemented. If any of the requirements are not feasible, the user is informed about this issue. A resolution is found and the user requirements document is edited accordingly.

Establishes the expectations for software functionality and identifies which system requirements the software affects. Requirements analysis includes determining interaction needed with other applications and databases, performance requirements, user interface requirements, and so on.

Design elements describe the desired software features in detail. The Software Design Document (SDD) contains the functional or dynamic design elements, such as business rules, business process diagrams. Physical

[7] The Unified Modeling Language is widely used method of visualizing and documenting software systems design, to represent the information system from a user's viewpoint.

Database Description (PDD) contains the static or structural design elements such as the entity relationship diagram, the access control matrix.

The software specification document which serves as a blueprint for the System development phase is generated in this phase. This document contains the general system organization, menu structures, data structure etc. It also hold system scenario, sample windows, reports for the better understanding. Other technical documents like entity diagram, data dictionary will also be produced in this phase.

When the SDD and PDD are finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement.

### 2.2.5 Specification of software requirements

All the technical specification is described in this phase. Which tools are required to develop this project? The need for the specific platform and technology is also decided in this phase of the development.

Specification is also made, if there is any need of other tools or technology to build the project.

### 2.2.6 Architectural design

The design of the Whole System is précised briefly in this phase. Whole system is divided in different modules or units. And documentation is prepared in this unit according to completion of the previous phase. Functionality of each unit is designed in this phase.

This phase determines the software framework of a system to meet the specific requirements. This design defines the major components and the interaction of those components, but it does not define the structure of each component. The external interfaces and tools used in the project can be determined by the designer.

This phase of the design of computer architecture and software architecture can also be referred to as high level design. The baseline in selecting the architecture is that in this phase list of modules, brief functionality of each

module, their interface relationship, dependencies, database tables, architecture diagrams, technology details etc are prepared.

### 2.2.7 Detailed design

In this phase all the units are designed in detail, as functionality of each unit is designed in previous phase. How this unit works, is designed in this phase? Which kind of functions should be there and functionality provided by that functions are decided in this phase. Characteristics of all the units are decided in this phase.

The designed system is broken up into smaller units or modules and each of them is explained so that the programmer can start coding directly. The low level design document or program specification will contain a detailed functional logic of the module or pseudo code.

Database tables with all elements, including their type and size are prepared in this phase. All dependency issues, error message list are prepared in this phase. Complete inputs and outputs are decided for a module.

### 2.2.8 Unit development

All the units are actually designed in this phase. This is the coding phase. Coding is being done in this unit. Units are developed under the technology, which is decided in previous phase. This is the actual phase where project is converted into reality.

This development stage takes as its inputs the design elements describes in the approved SDD and PDD. For each design element, a set of one or more software objects are produced. Software objects include but are not limited to menus, dialogs, and data entry forms, data reporting formats, and specialized procedures and functions. Appropriate test cases are developed for each set of functionally related software objects and an online help system is developed to guide users in their interactions with the software.

The RTM is updated to show that each developed object is linked to a specific design element and that each developed object has one or more

corresponding test case items. At this point, the RTM is in its final configuration.

The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an Implementation Map (IMP) that identifies the primary code entry points for all major system functions, a Software Test Plan (STP) that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM.

### 2.2.9 Unit testing

In computer programming, unit testing is a method by which individual units of source code are tested to determine if they are suitable for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. Unit tests are created by programmers or occasionally by white box testers. The main aim is to verify the internal logic code by testing every possible branch within the function, also known as test coverage. Static analysis tools are used to facilitate in this process, where variations of input data are passed to the function to test every possible case of execution. It is the main type of testing.

### 2.2.10 Software integration & testing

Integrated module is tested for their interoperability and functionality. In integration testing the separate modules will be tested together to expose faults in the interfaces and in the interaction between integrated components. Testing is usually black box as the code is not directly checked for errors.

During the integration and test phase, the software objects, online help and test data are transferred from the development environment to a separate test environment. At this point, all the cases run to verify the correctness and completeness of the software. Successful execution of the test proves a robust and complete migration capability.

During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Deployment Plan (DP).

The outputs of the integration and test phase include an integrated set of software, an online help system, an updated Implementation Map (IMP if necessary), a Deployment Plan (DP) that describes reference data and production users, an acceptance plan which contains the final document of test cases.

### 2.2.11 System integration & testing

System testing will compare the system specification against the actual system. After the integration test is completed, the next test level is the system test. System testing check if the integrated product works well and meets the desired requirements.

It is done because the customer who has ordered and paid for the system and the end user who will use the system may be different group of people or organizations with their own specific interests and requirements of the system. So software testing is done against technical specifications. And the system test is done to the system from the viewpoint of the customer and the future user. The tester confirms if the requirements are completely and appropriately met or not.

Many functions and system characteristics are outcome of the interaction of all system components, so they are only visible on the level of the entire system and can only be observed and tested there.

### 2.2.12 Installation at site

After whole system is integrated and tested, the project is installed at customer's place. Customer's place is visited to get the knowledge about the condition of the software or tools required to install the current system. After getting the information of the customer's system, all the applications which are required to run this project are also installed at the customer's place.

In this phase, all the required software to support or to run the developed system is installed properly at customer's site.

### 2.2.13 Site testing and acceptance

After installing the project at its customer's place, the whole project is tested again. The project is gone through "Unit testing". This testing is done as the unit is developed. When all the units are integrated then "Integrated testing" is done.

In this phase "User acceptance testing" is done. In this, testing is done by user, so the testing is done at user level. User will test the system on the basis of the demanded requirement.

If it works properly without any problem then it is accepted by the customer otherwise system is corrected to solve the problem.

### 2.2.14 Training and documentation

How the project work and how to operate the project, all this information is provided to the end user in training. Documentation of whole project is also provided to the end user for their future need. In the documentation all the required information related to the project is given, which help to the end user.

The documentation required to handle the system or user manual is also handed over to the customer. After installation of the system, end users are given the all over information about the system. So, that user can be familiar to the system and come to know how to use the system. Training is also provided with the operating information to the user.

In this phase all the information regarding the system is provided to the customer.

### 2.2.15 Implementation

This is the phase where the developed software is actually used by the end user. Whole the system is used for the task for which purpose that project is developed.

The user uses the newly developed system. Performance of the newly developed system is measured in this phase. So, the developers get the proper review for the system.

Actual use of the project is occurred in this phase of the SDLC life cycle model.

### 2.2.16 Maintenance

This phase is very important in SDLC model. This phase can be a project in and of itself. Future software upgrades, bug fixes, and regular maintenance are addressed during this phase. This phase may or may not have a well defined end state, as if user wants to see some changes in future. So it does keep on maintaining the software.

Maintaining the software is the endless process, which keeps on going till the software removal (which may or may not be there), as system upgrades or developed on the basis of the current system.

## 2.3 TYPE OF SOFTWARE DEVELOPMENT MODEL

There are many models an organisation can use as the basis of their approach to developing software. There are three main groups of development approaches under which a number of development models can be placed:

2.3.1 Unstructured or Ad-hoc

Big-bang, Code & Fix

2.3.2 Heavyweight or Predictive

Waterfall, V-Model

2.3.3 Lightweight, Adaptive or Agile

Spiral, XP, SCRUM

Brief description of the above models are describes below.

### 2.3.1 Unstructured or Ad-hoc

Development can be approached without any great attention to organising activities, resources or to the planning and analysis of what will actually be involved in delivering an item of development work.

An organisation can simply pour into a project the required elements of people, money and time then set off working in the hope that something useful will be produced. Alternatively they can enter a loop of finding and fixing issues until the product is deemed ‘good enough’.

This is the essence of unstructured and ad-hoc development approaches. Despite the apparently disorganized nature they can the best approach in certain circumstances.

#### *2.3.1.1 Bing-Bang Model*

The simplest form of development model is the Big Bang model. There is little in the way of formal process and the focus is on using the organisations energy to develop the final software. Defined phases, gateway criteria, documentation, testing, etc. are all considered non essential and are likely not to exist.

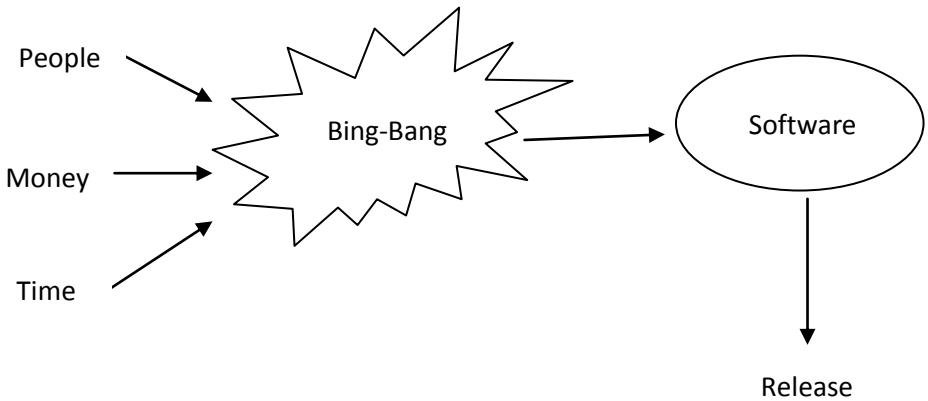



**FIGURE-4 BING-BANG MODEL**

This model does not allow for cyclic clarification of requirements, iteration of code, bug fixes or any other repeated activity as the norm today. Big Bang is a onetime only set of activities.

An organisation may say its development approach is Big Bang to describe the way development may send the test team a completed product for testing. Development are finished coding and ‘throw it over the fence’ to the test team who most likely have no previous knowledge of what’s been developed.

This is an erroneous use of the term Big Bang but as consultants it needs to understand why an organisation may think in these terms. Testing to find bugs and deliver fixes won’t be done in a strictly Big Bang approach, because if it is then the organisation has moved into a Code & Fix model.

### *2.3.1.2 Code & Fix Model*

A step forward from Big Bang would be to recognise that the product should be tested before release. This then sees the organisation delivering what they believe is completed code to the test team for testing.

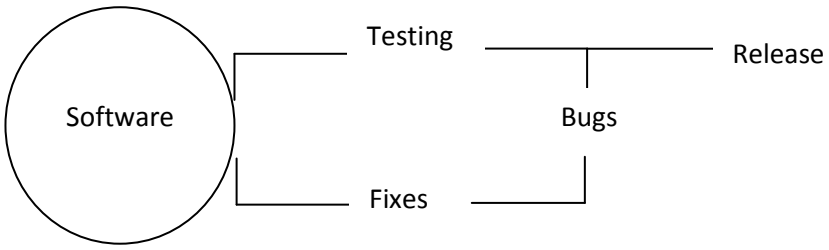



**FIGURE-5 CODE & FIX MODEL**

Code & Fix is often the default model within organisations that have not established a formal development approach. It’s important to realise that while Code & Fix is a model of itself it’s also a mode which organisations adopt at given phases of other models.

Whether an organisation is following Waterfall, Spiral or Agile they will all typically drop into Code & Fix at some point. It’s not uncommon to have three cycles of Code & Fix intentionally planned between builds, iterations or increments as a standard.

### 2.3.2 Heavyweight or Predictive Approach:

Heavyweight methodologies are considered to be the traditional way of developing software. These methodologies are based on a sequential series of steps, such as requirements definition, solution building, testing and deployment. Heavyweight methodologies require defining and documenting a stable set of requirements at the beginning of a project. There are many different heavyweight methodologies but I will limit our discussion to the three most significant methodologies: Waterfall, Spiral Model and Unified Process.

Some of the Heavyweight process are Waterfall Model, V-shaped Model, Spiral Model are discussed below.

### 2.3.3 Lightweight or Adaptive or Agile Approach

Agile as the name suggest “the quality of being agile that is readiness for motion, quickness, activity, handiness in motion”.

By this lightweight approach software development methods are attempting to offer once again an answer to the eager business community asking for lighter weight along with faster and quick software development process.

Some of the Lightweight processes are Scrum, Lean, Dynamic System Development, Rapid Application Development, Joint Application Development, and Extreme Programming.

All these methodologies acknowledge that high quality software and customer satisfaction could only be achieved by bringing lightness to their processes.

## 2.4 SOFTWARE DEVELOPMENT MODELS

### 2.4.1 Waterfall Model

The first formal description of the waterfall model is cited by Winston W. Royce[8]. The waterfall model is the classical model of the software engineering. This model is one of the oldest models and is widely used in government projects and in many major companies. As this model gives emphasis to planning in beginning stages, it guarantees design flaws before they develop. In addition, its intensive document and planning make it work good for projects in which quality control is a major aspect.

The waterfall development model originates in the manufacturing and construction industries; highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development[9].

Waterfall model of software development is separated into many separate phases and follows a sequential software development process. In a waterfall approach different phases are interrelated and come in a linear style. Each phase suggests a different task, which is completely dependent on previous phase.

The pure waterfall lifecycle contains several non overlapping stages, as shown in the following figure. The model starts with establishing system requirements and software requirements and continues with architectural design, detailed design, coding, testing, and maintenance. End of the one phase is the beginning of the other phase. Some certification mechanism has to be implemented at the end of the each phase. This is done by some verification and validation. Validation means confirming the output of a phase is consistent with its input and that output of the phase is consistent with overall requirements of the system.

[8] Royce introduced this model in 1970.
[9] http://en.wikipedia.org/wiki/Waterfall_model

The need of the certification is that each phase must have some defined output that can be evaluate and certified. Therefore, when the activities of a phase are completed, there should be an output product of that phase and the objective of a phase is to produce that product ultimately. The outputs of the earlier phases are often called intermediate products or design documents. For example, for coding phase, the output is the code. From this point of view, the output of a software project is to justify the final program or product along with the use of documentation with the requirements documents, design documents, project plan, test plan and test result.

The waterfall model considered as a baseline for many other lifecycle models. Waterfall model is a sequential design process, in which progress is steadily flowing downward.

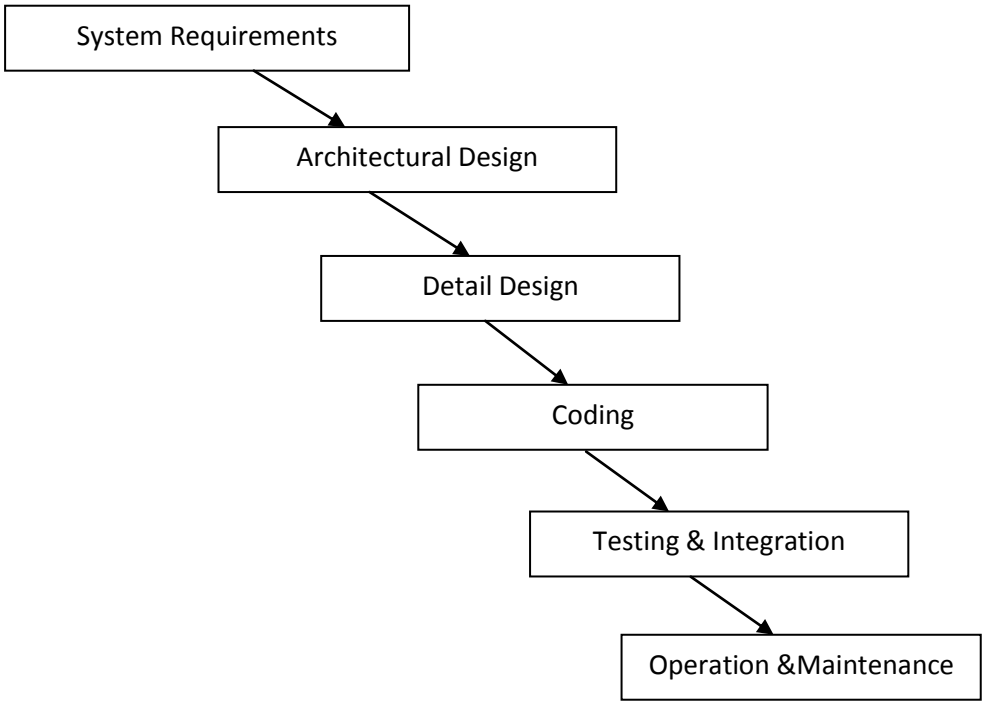



**FIGURE - 6 WATERFALL MODEL**

### *2.4.1.1 Phases of Waterfall Model*

**1) System requirements**

All the requirements of the system on which the system is going to be developed, are collected in this phase. Requirements have to be collected by analyzing the needs of the end user by doing questionnaires and having discussion to the user then checking them for validity and the possibility to implement them.

This phase defines that which components are going to be used for building the system, including the hardware requirements like hard disk space, display adapter, other peripherals like USB-port, pointing devices, network devices etc. software tools like which operation system is used and other necessary components like RAM, system's processing speed. In this phase, the needs of the customers are recognized and documented on a high abstraction level. Documents are prepared for the systems specific needs.

In this phase, establishes the expectations for system functionality and identifies which system requirements the software affects. Requirements analysis includes determining interaction needed with other applications and databases, performance requirements, user interface requirements, and so on. Requirements are divided in functional requirements, non functional requirements and constraints which the system has to fulfill.

System which is going to be developed should easily communicate with other application, as the system would be the part of a big system. It is also determine which database is suitable for the proposed system. How does the software behave, is also decided in this phase.

The aim is to generate a System Requirements Specification Document which is used as an input for the next phase of the model.

**2) Architectural design**

The proposed system has to be properly designed before any implementation is started. This includes an architectural design which defines and describes the main blocks and components of the system, their interfaces and

interactions. By this the needed hardware is defined and the software is divided in its components. E.g. this involves the definition or selection of a computer platform, an operating system, other peripheral hardware, etc. The software components have to be defined to meet the end user requirements and to meet the need of possible scalability of the system. The main objective of this phase is to produce a Software Architecture Document which serves as an input for the detailed design of the development, but also as an input for hardware design or selection activities. Usually in this phase various documents are generated, one for each discipline, so that the software usually will receive a software architecture document.

This phase decides the software framework of a system to meet up the specific requirements. This design defines the major components and the interaction of those components, but it does not define the detailed structure of each component. The external interfaces and tools used in the project can be determined by the designer.

**3) Detailed design**

The architecture design which defines the main software blocks of the system the software design will break them further down into code modules. The interfaces and interactions of the modules are decided in this phase, as well as their functional contents. All necessary system states like start up, shutdown, error conditions and diagnostic modes have to be considered and the activity and behavior of the software has to be defined. The output of this phase is a Software Design Document which is the base of the following implementation work.

So, study of the software components defined in the architectural design phase and produces a specification document for how each component is implemented is decided in this phase.

**4) Coding**

Based on the software design document the task is planned to set up the defined modules or units and actual coding is started. The system is first developed in smaller parts called units. They are able to stand alone from the

functional aspect and then they are integrated later on to structure the complete software package.

Implementation of the detailed design specification with specific tools is done in this phase. All the requirements puts into reality with this phase.

**5) Testing & Integration**

Each unit is developed independently and afterwards they can be tested for its functionality. This is the so called Unit Testing. It simply verifies if the modules or units to check if they meet their specifications and work efficiently. This includes functional tests at the interfaces of the modules, but also more detailed tests are done which consider the inner structure of the software modules. During integration the units which are developed and tested for their functionalities are brought together to build complete software package. The modules are integrated into a complete system and tested to check if all modules work in coordination as expected.

After successful integration of each module, including the related tests the complete system has to be tested against its initial requirements. This will include the original hardware and environment, whereas the last integration and testing phase may be performed in the predefined system environment or on a test module.

To check whether the software meets up the specified requirements and finds any errors is there in the code, software testing and system testing are done like this in this phase. Quality and functionality of the software are calculated in this phase.

**6) Operation & Maintenance**

The system is handed over to the customer and will be used the first time by him. Naturally the customer will check if his requirements were implemented as expected. In case there are changes necessary it has to be fixed to make the system usable or to make it to act in accordance with the customer wishes. In most of the "Waterfall Model" descriptions this phase is extended to a never ending phase of "Operations & Maintenance".

After the product has been released to the customer, it has to be maintained. If the customer found any problems in the product they report them to the company and get support in solving them. If the problems are due to the faults in the product, updated product is delivered to the customers.

### *2.4.1.2 Advantages of Waterfall Model*

1) Since the analysis team determines the business needs and requirements in starting, this process makes easy to better cope with the organizations need.
2) This process sets definite starting and ending points of a project.
3) As all the stages are clearly defined so, this process ensures early detection of errors and misunderstanding in its each stage.
4) Requirement specification document serve as the guideline for the development and testing phase.
5) In future, for code revision and future project enhancement these documents are useful in this process.
6) Since the following phases are dependent on previous phase, this approach ensures project deadline control.
7) Each phase is discrete and team members involved in a stage ensures the perfection of the stage before delivering to next stage. Waterfall process ensures greater project output.
8) This approach can be very efficient when team members are dispersed in different locations.
9) The amount of resources required to implement waterfall model is lower than other methods.
10) Since the output of prior phase is the input of the following phase, so each phase has to give some concrete result in terms of desired output.

### *2.4.1.3 Limitations of Waterfall Model*

1) The greater disadvantage of this approach is that there is no way to go back. Once a stage is complete means it is locked.
2) Sometimes it's very difficult to guess about time required for different phases and incorrect assumption can fail the project to meet up its deadline.
3) Waterfall model does not allow changes as per client's requirement, so it is less flexible.
4) If changes are to be made in waterfall process, the project has to be started all over again. This can be expensive for some organization.
5) Increase software development time and expense if customer keep on adding requirement to the list.
6) Team members of rest of the phase sits idle except the team members who are under the current working phase.
7) Poor model for long and ongoing projects.

### *2.4.1.4 Where to use the Waterfall Model*

When all the requirements are known at very early stage of System Development Life Cycle and unlikely to change in significant way during development and in the post-implementation period, in such situation Waterfall Model is used.

The system is so well understood by users that changes are not expected during the operation and maintenance.

Because of its weakness, application of the waterfall model should be limited to situation in which the requirements and the implementation of these requirements are very well understood.

The waterfall model performs well for product with a stable product definition and well understood technical methodologies. Software system design is not likely to undergo a change due to changes in technology, platform and language considered in the system.

If a company has experience in building a certain system like accounting, payroll, compilers then a project to build another of the same type of product perhaps even based on existing designs could make efficient use of the waterfall model.

In development of database-related software like commercial projects, in development of E-commerce website or portal, in Development of network protocol software this waterfall model can be used[10].

[10] http://www.ianswer4u.com/2011/11/advantages-and-disadvantages-of.html#axzz2bxfb2PI0

### *2.4.1.5 Modified Waterfall Model*

In this model, moving back in reverse is possible means developer can go back to the any phase from any phase. But necessary condition is that after reaching to the particular phase, one must have to passes through the all the preceding phases one by one. No jump is allowed in forward direction.

In all fairness, critics of this model must admit that the modified version of the waterfall model is far less rigid than the original, including iterations of phases, concurrent phase, and change management. Reverse arrows allow for iterations of activities within phase.

Although the modified waterfall is much more flexible than the classic waterfall model, it is still not better for rapid development project. As the requirements change the documentations would change. It would be confusing more and more as requirements keeps on changing. It will create messy environment for developer to develop large and rapid required project.

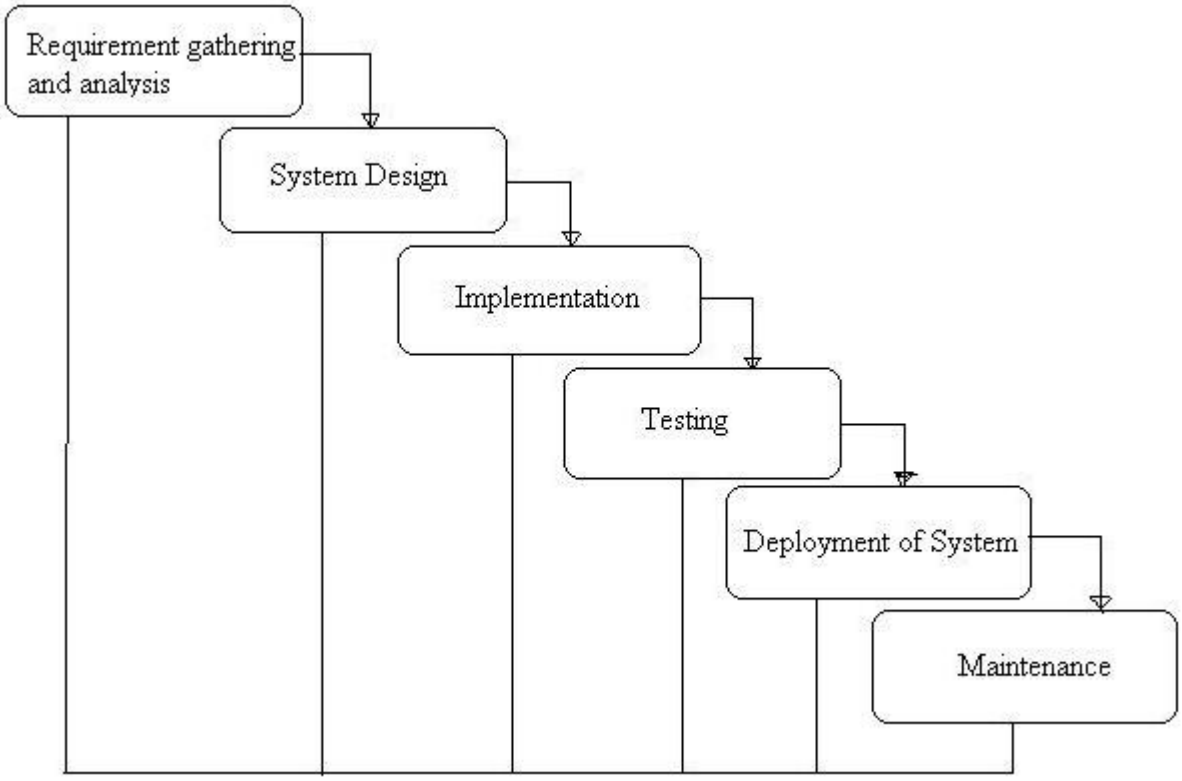



**FIGURE-7 MODIFIED WATERFALL MODEL**

### 2.4.2 Spiral Model

Waterfall model is one of the oldest and simplest models designed and followed during software development process. But this model has its own limitation such as error might drag to the next phase or identified at the next phase in the life cycle. All the errors or problems related to any phase are not resolved during the same phase, but those problems related to one phase are carried out in the next phase and are needed to be resolved in the next phase. This takes more time to solve the problem. This is the major risk factor which is the most important part, which affects the success rate of the software developed by the following Waterfall model.

In order to rise above the limitation of the Waterfall model, it was necessary to develop a new software development model, which could help in ensuring the success of a software project. One such model was developed that help to ease the software development process and eliminated almost every possible or known risk factors from it. This model is referred to as the “Spiral model”. This model is introduced by Boehm.[11]

The Spiral Life Cycle Model is type of iterative software development model which is normally implemented in high risk projects. In this system development method, features of both; waterfall and prototype models combine.

Spiral model is also called as meta-model because it consists of other models of SDLC. Both Waterfall and prototype models are used in it. Here software development is performed systematically over the loops and at the same time, in the same iteration prototype of the software is developed and shows it to the user after completion of various phases. This way risks can be reduced by using systematic approach.

There are four phases in this model which are: Planning, Evaluation, Risk Analysis and Engineering. These four phases are iteratively followed one after another in order to eliminate all the problems, which were faced in the waterfall model. Iterating the various phases helps to understand the problems with the particular phase and how to cope with those problems

[11] It is cited that this model is introduce in 1988 by Boehm.

when the same phase is repeated next time. Planning and developing strategies to be followed while iterating through the phases.

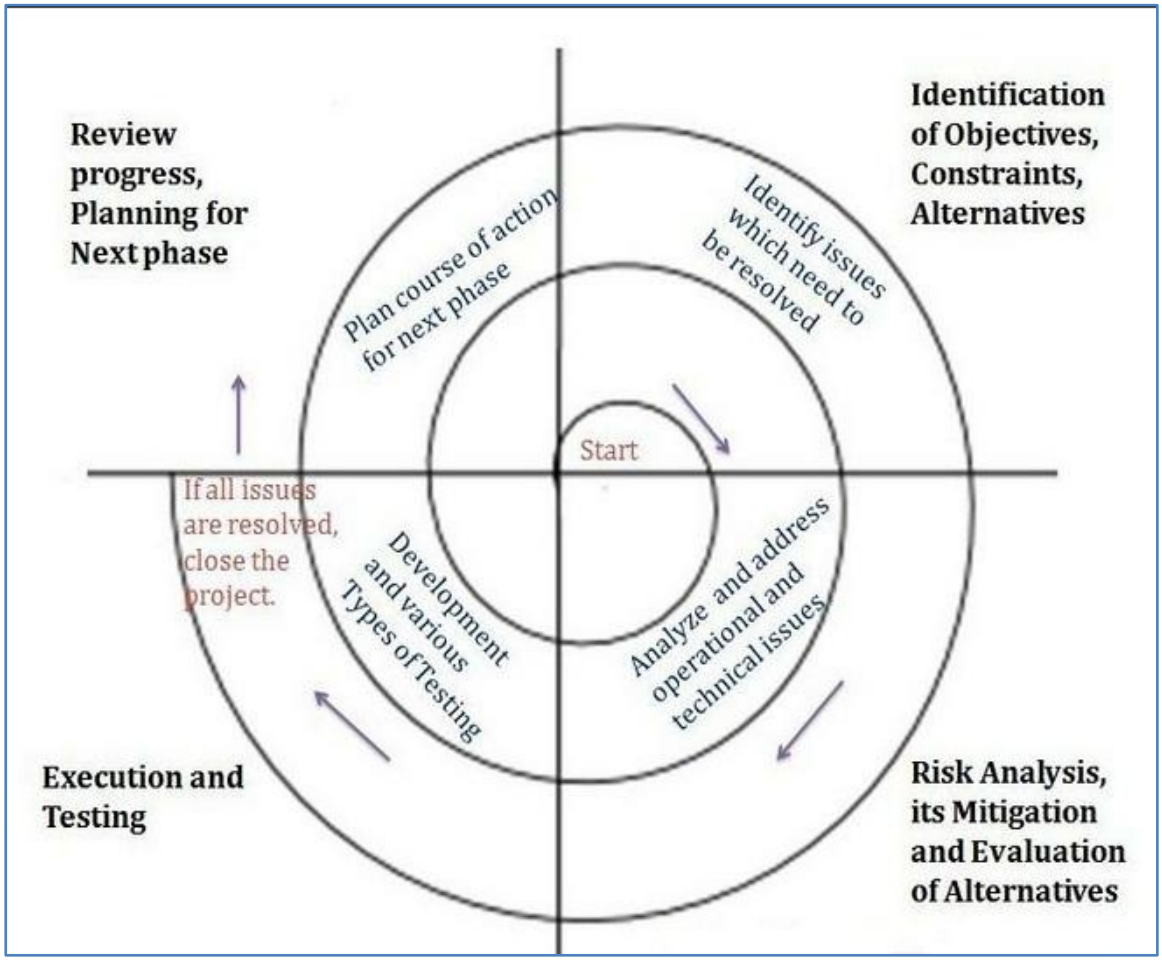



**FIGURE-8 SPIRAL MODEL**

### *2.4.2.1 Phases of Spiral Model*

**1) Plan**

This is the first phase of the model. In this phase, the goals or objectives, alternatives and constraints of the project are determined and are documented. The need and other criteria are decided in order to fix which approaches to follow during the project life cycle to achieve the decided goal.

It is needed that a lot of time must be invested for planning as it is the core of the spiral model. The simple carelessness would adversely affect the process badly.

**2) Risk Analysis**

This phase is the most important part of spiral model. This phase has been included specially in order to recognize and resolve all the possible risks in the project development. In this phase, all possible options, which can help in developing a cost-effective and good quality project, are analyzed and strategies are determined to use them throughout the life cycle of the project. If risks point out any kind of uncertainty in requirements, prototyping may be use to continue with the existing available data and find out a possible solution to cope with the potential changes in the requirements.

Operational and technical issues are dealt here. Risk lessening is the main focus in this phase. And evaluation of all these factors determines future action.

**3) Engineering**

In this phase, the actual development of the project is carried out. Coding of the project is done in this phase. So software is built in this phase. The output of this phase is passed through all the phases iteratively in order to get improvements in the software, in the actual project. Project is developed using waterfall model or incremental model approach. Testing is a part of this phase. All the units and integrated software is tested in this phase. The software system is tested during this phase. All the detected error or problem related coding is removed from software to make it bug free.

**4) Customer Evaluation**

In this phase, developed product is released to the customer to get feedback in terms of customer's comments and suggestions that can help in identifying and resolving potential problems or errors in the software developed. In this phase the testing of the developed software is being done again from customer's side. All kind of errors are identified in this phase and will rectify in the next prototype. Issues which need to be resolved are identified in this phase and necessary steps are taken to resolve them.

The process progresses in spiral manner which indicate the iterative path to be followed; increasingly more complete and more efficient software is built as we go on iterating through all four phases.

Successive loops of spiral model contain similar phases. Analysis and engineering efforts are applied in this model. Large, expensive or complicated projects use this type of life cycle. If at any point of time while developing, if risk is there in the project is more than expected then it should be eliminate it. Reviews at different phases can be done by an in-house person like team member of the development team or by an external client.

The first iteration in this model is considered as most important, as in first iteration almost all possible risk factors, constraints, needs are recognized and in the next iteration, all known strategies are used to build up complete software system. The radical dimensions shows evolution of the product towards complete system. All the activities are arranged in spiral model.

To follow Spiral model, highly skilled people in the area of planning, risk analysis, development, customer relation etc., are required. Process needs to be iterated more than once, demands more time and it would be an expensive task.

### *2.4.2.2 Advantages of Spiral Model*

1) Spiral life cycle model is one of the most flexible SDLC models. Development phases can be determined by the project manager, according to the complexity of the project.
2) Project monitoring is much easy and effective as it is done in each phase of each iteration. This monitoring is done by expert people or by concerned people. This makes the model more transparent and more effective to create successful project.
3) Risk management is core feature of this model, which makes it more useful compared to other models.
4) If changes are introduced at later stage in life cycle, coping with these changes isn't a very huge problem for the project manager to solve it.
5) It is suitable for high risk projects, where business needs may be unstable.
6) A highly customized product can be developed using this model.

7) Since the prototype building is done in small fragments or bits, cost estimation becomes easier and the customer can gain control on management of the new system by presenting their needs.
8) As the model keeps on moving towards final phase, the customer's expertise on new system grows, enabling smooth development of the product meeting client's needs.

### *2.4.2.3 Limitations of the Spiral Model*

1) Cost involved in this model is generally very high.
2) It is a complicated and time consuming approach.
3) Rules and protocols should be followed properly to successfully implement this model. Through-out the project life cycle development, it is very tough to follow all rules and protocols.
4) It is not suitable for low risk projects as cost would be high.
5) Meeting budgetary and scheduling requirements is very tough with this software development process as development flows in iterative manner.
6) Due to various customizations allowed from the client, it is very hard to use same prototype in other projects.
7) It needs extensive skill in evaluating uncertainties or risks associated with the project and their development.
8) The models work best for large projects only, where the costs involved are much higher and system pre requisites involves higher level of complexity.
9) Risk assessment expertise is required.
10) Spiral may continue indefinitely.

### *2.4.2.4 Where to use Spiral Model*

The spiral model is recommended where the requirements and solutions call for developing full-fledged, large, complex systems with a lot of features and facilities from scratch. It is used when new technology is there, trying out new skills, and when the user is not able to propose requirements in clear terms.

When requirements are not clear and when the solution has multi-user, multi-functions, multi-features where faultless integration, data migration are need in such scenario spiral model is use.

When cost and risk evaluation are concern at that time this process is useful. The project that has medium or high-risk, this model is useful. It is useful when requirements are uncertain and complex. The project that has long estimated time line then this model is used.

When the creation of prototype is the suitable type of product development, at the time when organization have the skills to tailor the model basis on which the software can be developed. During the progress in project important changes are expected.

At the time when completely new project is developed at that time success is not guaranteed. Organization can use this model that cannot afford to allocate all the money in the beginning of the project.

### 2.4.3 V-Shaped Model

V- Model means Verification and Validation model. The V-model represents a software development process which can be considered as an extension of the waterfall model[12]. It has inherited the same sequence structure of the waterfall model. Each following phase is begun at the completion of the deliverable of the prior phase.

This model is not moving in linear way, but the process steps are bent upwards after the coding phase which forms the typical V shape. The V-Model depicts the relationships between each phase of the development life cycle and its connected phase of testing.

The V-shaped model was built to help the project team in planning and designing for testability of a system. The model places a strong emphasis on the verification and validation activities of the product.

It illustrates that the testing of the product is discussed, designed and planned in the early phases of the development life cycle.

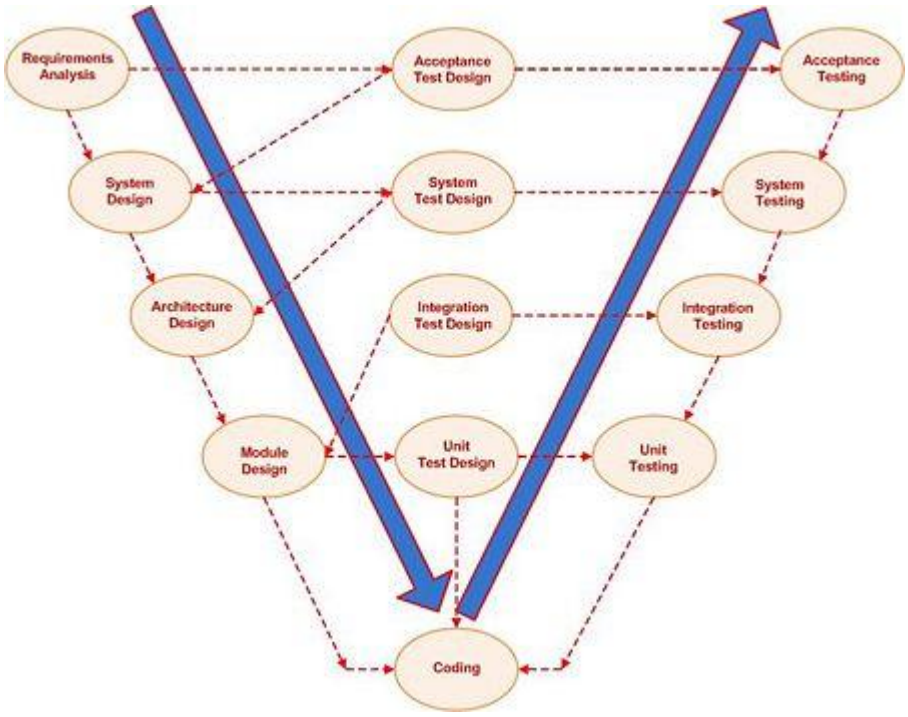



**FIGURE-9 V-SHAPES MODEL**

[12] http://en.wikipedia.org/wiki/V-Model_(software_development)

### *2.4.3.1 Phases of V-Shaped Model*[13]

**1) Requirement Analysis:**

In the Requirement analysis phase requirements of the proposed system are collected by analyzing the needs of the user. This phase decides the functionality of the proposed system. Usually, having discussion with the user a document called the user requirements document is generated.

The user requirements document describes almost all functionality of the system. It also describes interface, performance, data, security, etc. needs as projected by the user. It is used by business analysts to communicate their understanding of the system to the users. The user goes through this prepared requirements document as this document would serve as the guideline for the system designers in the system design phase. The user acceptance tests are planned in this phase.

**2) System Design**

System design is the phase where system engineers analyze and understand the business of the proposed system by studying the user requirements document. They plan out possibilities and techniques by which the user requirements can be implemented. If any of the requirements are not feasible, the user is informed about this issue. A resolution is found and the user requirements document is edited accordingly.

The software specification document which serves as a blueprint for the development phase is generated in this phase. This document contains the general system organization like which tools are used develop the software, menu structures, data structure etc. It also hold system scenario, sample windows, reports for the better understanding. Other technical documents like entity diagram, data dictionary will also be produced in this phase. The documents for system testing are prepared in this phase.

**3) Architecture Design (High level design)**

[13] http://en.wikipedia.org/wiki/V-Model_(software_development)

This phase of the design of computer architecture and software architecture can also be referred to as high level design. The baseline in selecting the architecture is that in this phase list of modules, brief functionality of each module, their interface relationship, dependencies, database tables, architecture diagrams, technology details etc are prepared.

The integration testing design is also carried out in this particular phase.

**4) Module Design (low level design)**

The module design phase can also be referred to as low-level design. The designed system is broken up into smaller pieces called units or modules and each of them is explained deeply so that the programmer can start coding directly. The low level design document or program specification will contain a detailed functional logic of the each module or pseudo code.

Database tables with all elements, including their type and size are prepared in this phase. All dependency issues, error message list are prepared in this phase. Complete inputs and outputs are decided for a module.

The unit test design is developed in this phase.

**5) Coding**

All the units are actually design in this phase. This is the coding phase. Coding is being done in this unit. Units are developed under the technology, which is decided in previous phase. This is the actual phase where project is converted into reality.

**6) Unit testing**

In this phase individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. Unit tests are done by programmer. The purpose behind this testing is to verify the internal logic code by testing every possible input within the function. It is to test every possible case of execution.

**7) Integration testing**

In this phase of process model the separate modules will be tested together to find errors in the interfaces and in the interactions among integrated units. Here code is not directly checked for errors but how all module work in cooperation when they are integrated is tested in this phase.

**8) System testing**

System testing will compare the system specification against the actual system. After the integration test is completed, the next test level is the system test. System testing check if the integrated product meets the desired requirements.

It is done because the customer who ordered and paid for the system and the end user who will use the system may be different group of people or organizations with their own specific interests and requirements of the system. So software testing is done against technical specifications. And the system test looks at the system from the viewpoint of the customer and the future user. The tester validate if the requirements are completely and appropriately met or not according to the system design document.

Many functions and system characteristics result from the interaction of all system components, so they are only visible on the level of the entire system and can only be observed and tested there.

**9) User Acceptance testing**

Acceptance testing is the phase of testing used to determine if a system satisfies the requirements precise in the requirements analysis phase. The acceptance test design is derived from the requirements document. The acceptance test phase is the phase used by the customer to determine whether to accept the system or not.

This testing helps to determine whether a system satisfies its acceptance criteria or not, customer would accept the system or not.

### *2.4.3.2 Advantages of the V-Shape Model*

1) Simple and easy to use as prior phases must be completed before starting following phases.
2) Testing activities are planned before coding. This saves a lot of time of development and also helps in developing a very good understanding of the project at the beginning state.
3) Each phase has specific deliverables so next phase can proceed smoothly.
4) Works well for where requirements are easily understood.
5) This model is work well for small projects.
6) Higher chances of success of this model because of the development of test plans are prepared early during the life cycle.
7) The model encourages verification and validation of all internal and external deliverables. The software product is also validated.
8) The V-shaped model encourages definition of the requirements before designing the system and it encourages designing the software before building the components. Test plan is also done with all the ongoing phase.
9) It defines the deliverable that the development process is going to generate, each deliverable must giving some concrete output to the next phase.

### *2.4.3.3 Limitations of the V-Shape Model*

1) It is inflexible; it has no ability to respond to change. It is very rigid as there is no way to go back.
2) It produces inefficient testing methodologies if any changes come across during development.
3) If any changes happen in mid way, then the best documents along with requirement document has to be updated.
4) Adjusting requirements is difficult and expensive.

5) Software is developed in the implementations phase, so no prior prototypes of the software are produced.
6) Model doesn't provide a clear path for solving problems found during testing phases.
7) It doesn't handle concurrent event as it is extension of waterfall model.

### *2.4.3.4 Where to use V-Shape Model*

Where time and cost is the main factor of the project then we can use such models for quick and cost effective delivery. In comparison with waterfall model more or less same but the activity of testing starts very early, which leads to less time, and cost of the project.

Like its predecessor, the waterfall model, the V-shaped model works best when all knowledge of requirements is available in the beginning.

This works best when knowledge of how to implement the solution is available, technology already exists and all the staff has expertise and experience with the technology.

The V-shaped model is the best choice for systems that require high reliability, such as hospital patient control application and embedded software for air-bag controller in automobiles due to their provision of test plan which produce test cases in all the phase of the development.

The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed. The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.[14]

[14] http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/

### 2.4.4 Rational Unified Process[15]

The Rational Unified Process is a Software Engineering Process. It provides a more accurate approach to assigning task and responsibilities within a development organization. It ensures the production of high-quality software that meets up the requirement of its end-users, within a defined schedule and cost or budget. It is published by Ivar Jacobson, Grady Booch, and James Rumbaugh[16].

The Rational Unified Process is a process product. The development team for the Rational Unified Process is working closely with customers, partners, Rationale's product groups as well as Rationale's consultant organizations, to guarantee that the process is constantly updated and changed upon to reflect recent experiences and growing and proven best practices.

The RUP increases team output, by providing every team member an easy access to knowledge base guidelines, templates and tools for all critical development activities. By having all team members accessing the same knowledge base, no matter if team members work with needs, design, test, project management, or configuration management. All the team members share a common language, process and view of how to develop software.

The Rational Unified Process activities build and maintain model. The Unified process emphasizes on the development and maintenance of models, rather than focusing on the production of large amount of paper documents.

The RUP is a guide for efficiently using Unified Modeling Language. The UML is the efficient language which allows to clearly communicating requirements, architecture and designs.

The RUP is a configurable process which can be customizing according to customer's needs. Single process is not suitable for all software development. The RUP is fits for small as well as large development organizations. It

[15] http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[16] It is cited that this model is introduced in 1999 by Ivar Jacobson, Grady Booch, and James Rumbaugh.

contains a Development kit, providing support for configuring the process to fit the needs to accommodate different situations.

There are six main basic fundamental practices of the Rational Unified Process.

1. Develop software iteratively
2. Manage requirements
3. Use component-based architectures
4. Visually model software
5. Verify software quality
6. Control changes to software

### *2.4.4.1 Basic Practises*

**Develop software iteratively**

In today's era, it is not possible to do all the process sequentially, first define the entire problem, design the entire solution, build the software and then test the product at the end. An iterative approach is needed that performs an increasing understanding of the problem through successive refinements, and to incrementally grow a successful solution over multiple numbers of iterations. The Rational Unified Process supports an iterative approach to development that addresses the highest risk items at every stage in the lifecycle, to reduce a project's risk profile. This iterative approach helps in reducing risk through frequent iteration, executable releases which includes continuous end user participation and feedback. Each iteration ends with an executable release, the development team stays focused on producing results, and frequent status checks help ensure that the project stays on schedule. An iterative approach also makes it easier to accommodate changes in requirements, features or schedule if necessary.

**Manage Requirements**

The Rational Unified Process describes how to discover, organize, and document required functionality and constraints tradeoffs and decisions. Easily collected and communicate business requirements. The use of use case and scenarios controlled in the process is explained as the best way to capture functional requirements and to guarantee that these drive the design, implements and testing of software. Here, requirements are prioritized according to the customer need. Next iteration provides some more facility or meets the needs of the customer. It could be made in such a way that the final system fulfils the end user needs. Management of requirements provides logical and visible way to the development and to deliver system.

**Use Component based Architecture**

The process describes to design flexible and useful architecture that accommodates changes, which is easily understandable and encourages more effective software reuse. The RUP supports component based software development. Components are modules, subsystems that fulfill a clear function. The RUP gives an organized approach to define structure using new and existing components.

**Visually Model Software**

This process shows you how to visually model software to build structure and behavior of architectures and components. UML is used to implement visual modeling. By using UML one can see how the elements of the system fit together; all the component of the system communicate with each other properly. It will look after those building blocks are consistent with your code; sustain consistency between a design and its implementation providing smooth communication between components.

**Verify Software Quality**

Poor application performance and poor reliability would be the factors of failure of today's software applications. So quality should be reviewed with respect to the requirements based on reliability, functionality, application performance and system performance. The RUP helps in the planning, design, implementation, execution and evaluation of these test types. Quality assessment is built into the process, in all activities, involving all team

members, using objective measurements. It can be done as separate activity performed by a separate group.

**Control Changes to Software**

In current environment changes are unavoidable. The ability to manage change is making in such a way that each change is acceptable, and software process has to able to cope up with changes. This process explains how to control, track and monitor changes to enable successful iterative development. It also direct how to set up secure workspace for each developer by providing separation from changes made in other workspace and by controlling changes of all software model. And it gets a team together to work as a single unit by describing how to automate integration and build management.

**Two dimensions of the Process**

The process can be described in two dimensions.

The horizontal axis represents time and shows the dynamic aspect of the process as it is performed and it is expressed in terms of cycles, phases, iterations and milestones.

The vertical axis represents the static aspect of the process: how it is described in terms of activities, objects, workers and workflows.

### *2.4.4.2 Phases of Rational Unified Process*

The software lifecycle is broken down into cycles, each cycle working on a new generation of the product. The RUP divides one development cycle in four consecutive phases.

- Inception phase
- Elaboration phase
- Construction phase

- Transition phase

Each phase is concluded with well defined milestone – a point in time at which certain critical decisions must be made, so that key goals could be achieved.

Each phase has specific purpose.

**1) Inception phase**

In the inception phase, business case for the system is established in this phase. To achieve this, all scenarios are inspected and information would be gathered to which the system will interact. By using use cases requirements are collected. This phase includes success criteria, risk assessment and estimate of the resources needed and plan showing dates of major milestones.
A wide vision of the core project's requirements, core key features of the software and core constraints is decided in this phase. Initial project requirements are decided in this phase.
An initial business case is established, which includes business context, initial risk assessment, market recognition, financial forecast, completion time line.

By end of this phase stakeholder makes agreement on scope definition and estimate the cost and schedule of the project. Requirements understanding are done by primary use cases. By doing these reliable requirements are collected. Cost and risk assessment is done in this phase.

**2) Elaboration Phase**

The purpose of the elaboration phase is to analyze the problem domain in depth, establish a sound architectural foundation, develop the project plan, and eliminate highest risk elements of the project. Architectural decision is made with an understanding of the whole system: its scope, major functional requirement and non functional requirements.

At the end of the phase, the project goes through its most important part of estimation: the decision on whether or not to hand over to the construction and transition phases. The process must accommodate changes, the elaboration phase activities must guarantee that the architecture of the system, requirements and plans are steady enough, and the risks are minimized. So, one can determine the cost and schedule for the completion of the development.

In this elaboration phase, an executable architecture prototype is built in one or more iterations, depending on the scope, size, risk and originality of the project. This effort should identified major technical risks of the project by using use cases.

In this phase, risk assessment is done again and technical risks of the project are identified if it is there and revised risk list is prepared and a revised business case is also prepared. A development plan for the complete project and evaluation criteria for iteration is also prepared. A preliminary user manual is also prepared.

Actual decision is taken here that the project may be aborted or continue if something goes wrong with respect to size, scope, and cost or with any aspects of the project development.

**3) Construction Phase**

In the construction phase, all the components and application features are developed and integrated into a single product and all features of the product are tested. The construction phase is a manufacturing process where importance is places on managing resources and controlling operation to minimize cost, schedules, and quality. Many projects are large enough that parallel constructions are being done. These parallel activities can speed up the deployable release, which can increase the complexity of resource management and workflow synchronization. A robust architecture and an understandable plan are highly correlated. This is the reason why the balanced development of the architecture and the plan is stressed during the elaboration phase.

At the end of the construction phase a product is ready to hand over to customer to use that.

At minimum, it consists of:

- The software product integrated on the required platforms.
- The user manuals.
- A description of the current release.

At this point, if the software, the sites, and the users are to use the software, without exposing the project to high risks. This release is often called a “beta” release.
Next phase may have to be postponed by one release if the project fails to reach this stage.

**4) Transition Phase:**

The purpose of the transition phase is to release the software product to the user. Once the product has been handed over to the end user, issues usually arise that require developing new releases, correcting some problems, or finishing the features which were postponed during the development.

The transition phase is entered when software is developed and efficient enough to be deployed in the end-user system. This usually requires that some usable part of the system has been completed to an acceptable level of quality which would be deployed at the user side. User documentation is available so that the transition to the user will provide positive results for all stakeholders.

- “Beta testing” to validate the new system against user expectations.
- Parallel operation with inherited system that it is replacing.
- Conversion of operational databases.
- Give training to users and maintainers.

- Roll-out the product to the marketing, distribution, and sales teams.

All this activities are done in this phase.

The transition phase focuses on the activities required to hand over the software into the hands of the users. Typically, this phase includes several iterations, including beta releases, general availability releases, and bug-fix. Significant effort is expended in developing user-oriented documentation, training users, supporting users in their initial product use, and reacting to user feedback. At this point in the lifecycle, however, user feedback should be restricted mainly to product modification, configuring, installation, and usability issues.

The primary objectives of the transition phase are achieving user self-supportability, achieving stakeholder agreement that deployment baselines are complete and consistent with the evaluation criteria of the software.

To achieve final product as rapidly and cost effectively as practical this model is useful. This phase can range from being very simple to extremely complex, depending on the type of product.

For example, a new release of an existing desktop product may be very simple, whereas replacing a nation's air-traffic control system would be very complex.

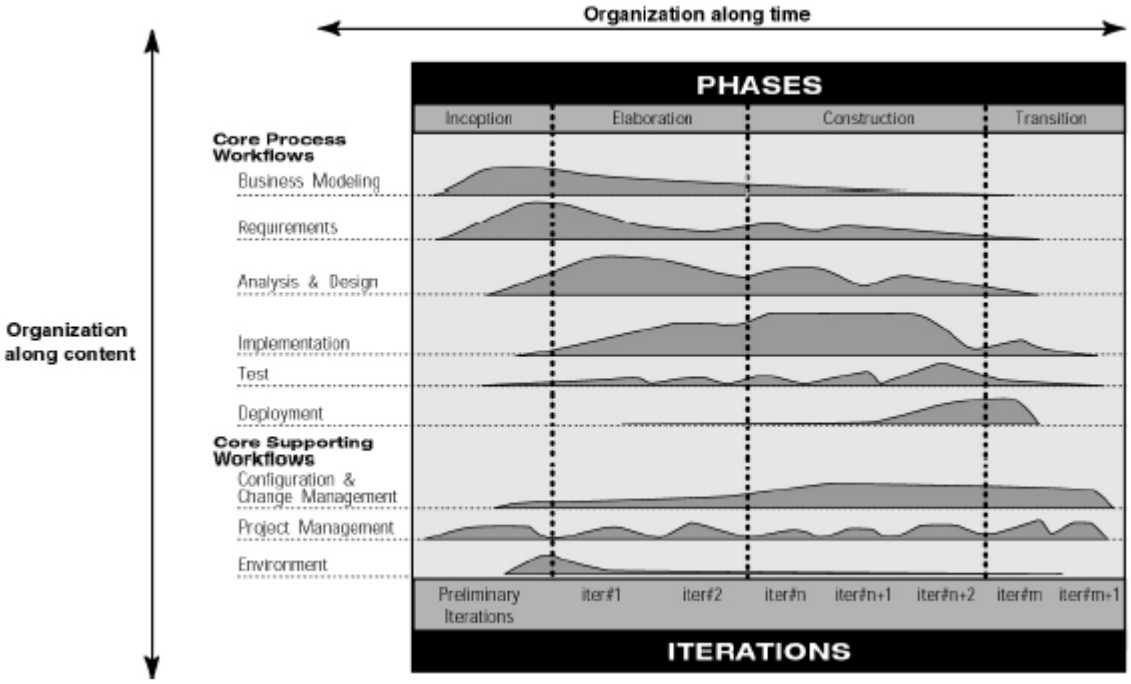



**FIGURE-10 RATIONAL UNIFIED PROCESS**

### *2.4.4.3 Advantages of the Rational Unified Process*

1. RUP is a complete methodology to manufacture software which is working efficiently.
2. It is process with complete document facility provided during development process which would also helpful in future.
3. It supports changing requirements to meet up its desired software.
4. It supports iteration process so we can integrate the code in development life cycle in lesser time and effort spent in integration.
5. The reuse of code easy and faster so development time is less.
6. There is online training and tutorial available for this process.
7. Debugging is very easy due to component base architecture.
8. The software is developed by this model can be enhanced further easily due to component based architecture.

### *2.4.4.4 Limitations of the Rational Unified Process*

1. The team members need to be expert in their field to develop the software under this methodology.
2. The development process is too complex and disorganized.
3. On cutting edge projects which utilize new technology, the reuse of components will not be possible. Hence the time saving one could have made will be impossible to fulfill.
4. Integration throughout the process of software development, in theory sounds a good thing. But on particularly big projects with multiple development flow it will only add confusion to the integration and cause more issues during the stages of testing.
5. It's too complex to implement, and too difficult to learn.
6. It may lead to undisciplined form of software development.

### *2.4.4.5 Where to use Rational Unified Process*

RUP is appropriate for the software project for wider scope. RUP is used to develop project when experts are there within the software development team. When project is small then this RUP is used and it would be very simple. But when project is vide in scope then using RUP would be very complex but gives very perfect output for this kind of project.

RUP is based on component based nature so it allows the system to be debugged and changed in parts. So, when changes are constantly made during the project then this model is used.

Documentation are the best part of the model so, when strong emphasis on documentation for further studies or requirements then this model is used. As, this model uses iterative approach and incremental approach. To develop next generation of the current system, document which prepared during last increment is used.

### 2.4.5 Prototype Model

The prototype model was developed on the assumption that it is very difficult to know all the requirements at the initial stage of a project. Users know many of the objective that they wish to have with their system but they don't know about the data, nor they know the details of the system features and functionality.

The prototype model permits for these conditions and offers a development approach that gives results without prior requiring all information in the beginning.

As the name suggest Prototype is develop initially and on basis of this final product is developed. It is early approximation of final product. A prototype acts as sample model, from this final product is developed.

With using the Prototype Model, the developer develops a simplified version of the proposed system and displays it to the customer for consideration of that system, as a part of the development process. The customer gives the feedback by using the system, which are in the form of requirements. Again developers consider these new requirements and new system is developed based on new requirements.

The basic idea here is that instead of finalizing the requirements before a design or coding can proceed, a prototype is built to make clear the requirements. This prototype is developed based on the currently known requirements. Prototype development goes through all the phases like design, coding and testing. But each of these phases is not done properly or thoroughly. But by using this prototype, the client or customer can get an actual idea of the system and they can make able the client to understand the requirements of the desired system better. So while making the model time to time feedback is given to the developer by the customer about their requirements. Completely built sample model is shown to user and based on his feedback; the SRS (System Requirement Specification) document is prepared.

After completion of this, more specific SRS document is prepared and now final product is prepared by using Waterfall Model.

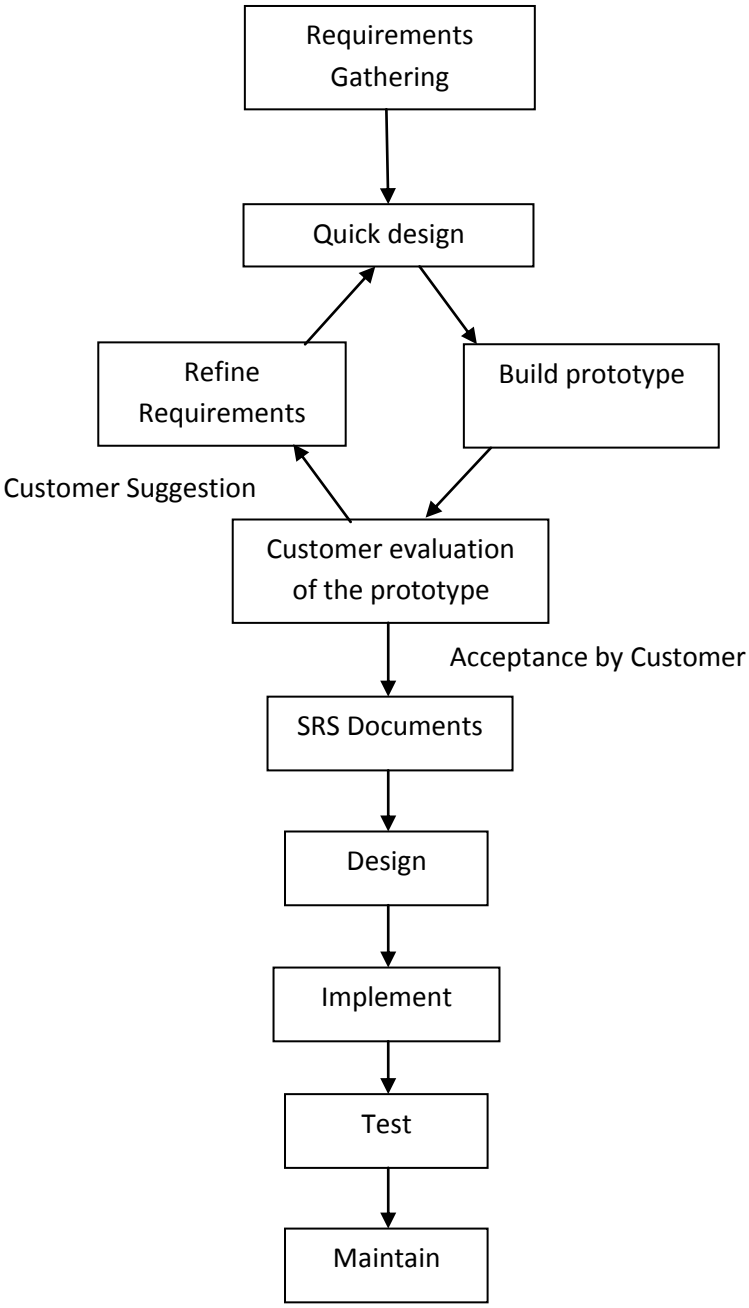



**FIGURE-11 PROTOTYPE MODEL**[17]

[17] http://verification-validation.blogspot.in/2012/07/softwarelife-cycle-models.html

### *2.4.5.1 Phases of Prototype Model*

**1) Requirement Gathering**

This is the phase where requirements are gathered during this phase. All the possible requirements are handed over to the developers. The information collected is usually subset of the complete system requirements.

All the functional requirements and constraints are defined in this phase. This is the phase where system requirements and software requirements are collected together. The behavior of the software, what type of need is there, how the software interacts with other applications are defined in this phase. Tools required to build the software are also defined in this phase.

Based on these requirements the prototype model is created.

**2) Quick Design**

Design is created based on the requirements gathered during requirement gathering phase. This design will be used further for next phase for the development. The technology which is going to be used in the system is decided in this phase.

The hardware and software to build the software are decided in this phase. The database which is going to use is also decided in this phase. The template of the system comes into existence in this phase. On basis of this template the actual software is prepared in the next phase.

In this phase, the whole system is divided in the small units also. The behavior of these units is also decided in this phase.

**3) Build Prototype**

Actual model is prepared in this phase. Coding is done in this phase. Whole system is prepared in this phase. First all the small units called units are prepared in this phase. These units are tested under “Unit test”. Then all the units are integrated and “Integrated test” is being performed. This model works as prototype model. It designed under the given requirements.

**4) Customer evaluation of Prototype**

After using the prototype model, customer gives their feedback. Customer evaluates this software on the basis of their requirements. If their requirements are fulfilled and prepared prototype works as their needs then the prototype would accepted. And if prepared prototype is lacking in some requirements then customer also gives suggestion for more requirements and changes.

**5) Refine Requirements**

Newly arrived requirements are processed in this phase, which is the base for new system to develop. Again all the requirements are analyzed and prioritized them in order to fulfill them in a particular order.

Phases no 2, 3, 4, and 5 are repeated until required design model is prepared. It displays to the client and if it is approved then related document is prepared.

**6) SRS Documents**

System Requirement Specification (SRS) documents are prepared, which describe the whole system requirements. This is the document by reading this one can find the actual requirements of the system. So, it would be very easy that if any team member is not present at work then another team member easily identify what to do next.

**7) Design**

Depending upon SRS document the required model is built. Again design template is getting ready. It would be now easy to design the system; all the requirements are described in the SRS documents. Again testing at developer side is done in this phase. If any error is there, then again it needs correction. After making it error free now, it is ready to use.

**8) Implement**

In this phase the software is ready to implement at client's site. Now actual use of the software is being made.

To implement at customer's site, it is to check that customer's computer system have the platform which is required for the newly developed software. If there is not such tools and environment available then it must be installed first; like required hardware tools, software tools, operating system, database, etc.

**9) Test**

At client's site, the software is tested for its functionality, correctness and interoperability. Software which is developed is working properly and gives the result as per need of the customer. Of course interaction of the developed software with other application is check in this phase.

It is tested as per user requirements. So the software meets the desired requirements or not, user can find out.

**10) Maintain**

This is the lifelong phase. Software is maintained for any error in future. By using this system another system also can be developed.

### *2.4.5.2 Advantages of the Prototype Model*

1) When prototype Model is shown to the user, he gets a proper clarity about his requirements. And know the functionality of the software, so can suggest the changes and modifications.
2) This type of approach of developing the software is used for non-IT literate people. They usually cannot explain their requirements specifically.
3) When client is not confident about the developer's capabilities, he asks for a small prototype to be built. Based on this prototype model, he can judge capabilities of developer.
4) It helps to demonstrate the concept to the investors to get funding for project.
5) It reduces risk of failure, as potential risks can be identified early and steps can be taken to remove that risk.

6) Constant interaction between development team and client provides a very good and co-operative environment during project.
7) Time required to complete the project after getting final SRS is reduces as the developers has a better idea about how he should approach the project.
8) The customer does not need to wait long for working software.
9) Feedbacks from customer are received periodically and the changes don't come as a last minute surprise.

### *2.4.5.3 Limitations of the Prototype Model*

1) Sometimes the start-up cost of building the development team, focused on making prototype is high. And usually in starting Prototype is done at the cost of the developer. So it is done using minimal resources.
2) It is a slow process.
3) Once we get proper requirements from client after showing prototype model, it may be of no use.
4) Too much involvement of client is not always preferred by the developer.
5) Too many changes can disturb the development team.
6) Customer could believe the prototype as the working version.
7) Developer also could make the implementation compromises where he could make the quick fixes to the prototype and make is as a working version.
8) Often clients expect that a few minor changes to the prototype will more than suffice their needs. They fail to realize that no consideration was given to the overall quality of the software in the rush to develop the prototype.

### *2.4.5.4 Where to use Prototype Model*

When the client has only general view of what is expected from the software product. In such scenario where there is an absence of detailed information regarding the input to the system, the processing needs and the output requirements, the prototyping model may be employed. This model reflects an attempt to increase the flexibility of the development process by allowing client to interact and experiment with a working representation of the product.

Prototyping used for large and complicated system for which there is no manual process or existing system to help to determine the requirements and when high risk is associated with the system.

Prototype model is useful or used when complete new project is developed or the project is not inherited from any previous unit or system.

When high technology system is developed where requirements are generalized, they are not specifically identified at that time this model is used.

Prototype model is used for user interface like interactive online system, decision support system such as medical diagnosis system.

It[18] has been found that prototyping is very effective in the analysis and design of on-line systems, especially for transaction processing, where the use of screen dialogs is much more in evidence. The greater the interaction between the computer and the user, the greater the benefit is that can be obtained from building a quick system and letting the user play with it

[18] http://en.wikipedia.org/wiki/Software_prototyping

### 2.4.6 Iterative and Incremental Model

The term ‘iterative’ indicates the repetition of one or more activities; the term ‘incremental’ means that the development proceeds from an initial subset of the requirements to more and more complete subsets, until the complete system is built. Incremental development means step wise create the partial system of the proposed system functionality or intended system functionality. Some or all of the succeeding development activities can be carried out independently and in parallel. The ultimate system results from the total integration of the partitions.

In iterative and incremental processes there is still a need for analysis, design, implementation and testing activities, but these activities are carried out in a more flexible manner than in the waterfall process. An iterative and incremental process made of several cycle of analysis, design, implementation and testing. Each cycle is short and supply feedback for the next cycle, in which a more refined and more enhanced development model is achieved. With an incremental model, development starts from small subset of requirements, lessening the complexity and scope of each analysis, design and coding cycle. Each increment is carried through the development activities to generate a working portion or subset of the system, and is developed through several iterations. The integration of the increments results in the final system. This integration can be increasingly completed by successive releases of the software, each release achieve more functionality and moving towards the complete system which meet up to the all requirement of the user.

As shown in below figure, the increments represent development that can be carried out independently and in parallel.

In starting, the linear sequence of the waterfall model is applied until one increments built; this process is then repeated until the next increment of the software is delivered, these processes can be conducted in parallel. The problem would have to be divided into several sub-problems so that can be developed in turn. Normally there is a practise to prioritise requirements initially, and those high priority requirements are usually incorporated into

initial increments. As each partition becomes complete it is tested and integrated with other partitions. Although system requirements are fixed for each increment, the system can be evaluated at each stage to decide for future partitions, which permits constant progress in completion of requirements.

The reason of iterative developments is to allow reworking on part of a system to remove mistakes or to make improvements based on user feedback. After iteration, an evaluation of the result and planning to improve the correctness of the system and the quality of design would need to happen. The number of iterations that would need to occur could be uncertain depending on how much of the system would need to be corrected; so, it may be difficult to set an end to the project. But it can go as long as required system is not developed.

There are four steps, which are followed by incremental development.

### *2.4.6.1 Phases of Incremental and Iterative Development*

**1) Analysis**

In this phase, requirements for supposed system are analysed. Information related to the requirements is collected. Feasibility studies are being done under this phase. All kind of feasibility study had been done studied under this phase.

Basic requirements are gathered for first step of incremental phase of first iterative phase. Then after completing all the phases for first iteration, information collected during first iteration will be the information for second iterations. Like, after completion of all the iteration, all the information collected during that iteration will be the first phase for the next iteration.

**2) Design**

During this phase, architectural design and detailed design for the supposed system is prepared. Functionality of the different parts of the system is decided. How system will work is decide in this phase.

In architectural design, the software and hardware which are going to be used to develop the system are decided. The database as backend tool is also suggested in this phase.

In detailed design the while system is divided in the small units and the functionality of the units are also decided in this phase.

How the system work, the behaviour of the system with other application is also decided in this phase.

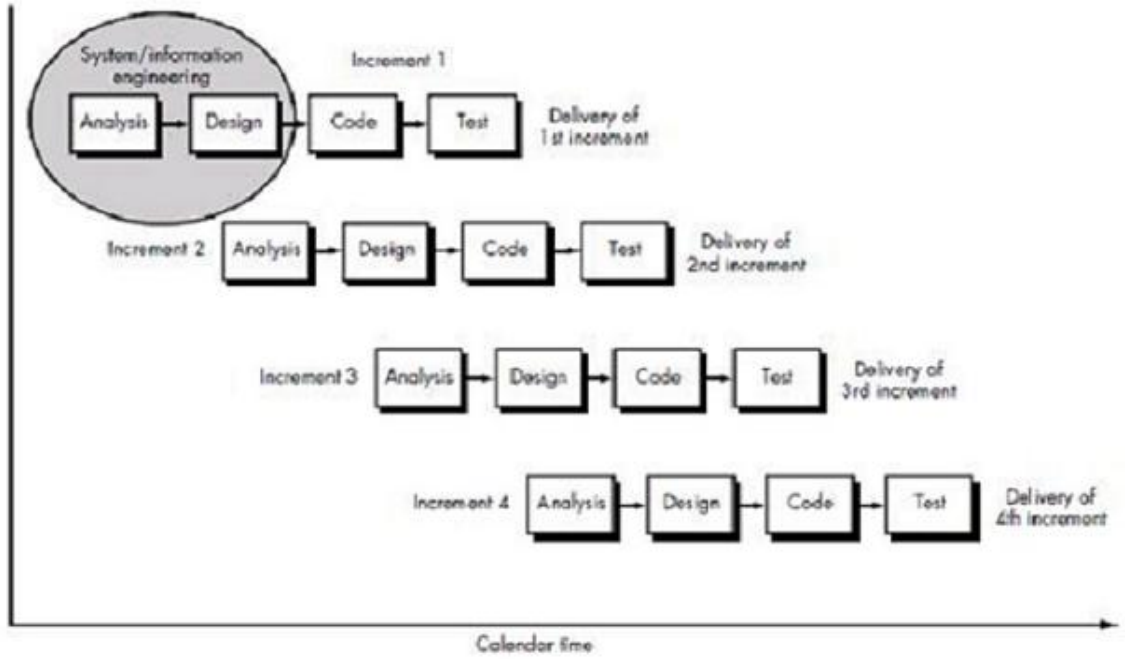



**FIGURE-12 ITERATIVE AND INCREMENTAL MODEL**

**3) Code**

Actual implementation of the system converts into coding with the help of required programming language, which is decided in the analysis phase. This is the phase where system design comes to the real software. All the units build in this phase.

**4) Test**

Testing is done in this phase. All the errors or problems are eliminated in this phase. All the units are tested by “Unit testing” and after successfully completed this testing all the units are integrated and tested under “Integrated

testing". This is the most important phase. After completing this phase, the software system is hand over to the customer.

These phases are repeated continuously to achieve the desired software system. After completion of iteration, software reaches more nearer to the desired system means iterations are continue till the desired software is built.

This model is used when requirements are clear but project scope requires pure linear approach. In this model first project is deliver to customer is a core project after this we give other features of the software. We get feedback from customer after being checked core product if there is any flaw in the core product then it is removed in next increment. we require only few staff if whole staff is not present then we can start with core product or some features and deliver to customer until other staff members were come and join in next increment. In this model we can manage the technical risks also.

### *2.4.6.2 Advantages of the Iterative and Incremental Model*

1) Versions of the proposed system are provided after iteration of the incremental model.
2) After using first iterated model, user can give their suggestion and demand for changes.
3) This model does not affect anyone's business values because they provides core of the software which customer needs after first iteration. To the customer software will help him to keep running his business.
4) It is flexible to the customer's requirements and easy to manage model.
5) Better risk management is there in this model because one can confirm the outcome by the customer after every version because every version is prepared according to the plan.
6) Easy to test as testing is done in iteration as per requirement.
7) This model is used when requirements are clear to some extend but project development requires pure linear approach.
8) Complete implementation is there by decided dead line.
9) Sometimes early increments can be implemented with fewer people.

10) Lower risk of project failure compared to other approaches.
11) Results are obtained early and periodically.
12) Parallel development can be planned.
13) Progress can be measured by setting milestone.
14) Testing and debugging during smaller iteration is easy.

### *2.4.6.3 Limitations of the Iterative and Incremental Model*

1) Each phase of an iteration is very rigid and do not overlap each other.
2) Problems may arise related to system architecture because not all requirements are gather in initial stage of the development process.
3) Each increment needs to be relatively small.
4) Mapping requirements to increments may not be easy so managing documents are very difficult.
5) Common features of the software are difficult to identify because of continuous changes in requirements.
6) During development process changes are being done at first iteration. As if it continues to change and it never finished.
7) More management attention is required due to frequent changes in requirements.
8) Does not allow iterations within an increment.

### *2.4.6.4 Where to use Iterative and Incremental Model*

This model is used where requirements are clear or almost clear and can implement by phase wise. Mostly such model is used in web applications and product based companies.

Such models are used where requirements are clear and can implement by phase wise. Mostly such model is used in web applications and product based companies[19]. This model is used for large system which built in small phases or segments. Also it can be used in system has separated units, for i.e. ERP system, which we can start with budget module as first iteration and then we can start with inventory module and so.

This model is used when there is a requirement to get basic functionality to the market quickly.

This model is used when new technology is used within the project, allowing the user to adjust to the system in smaller incremental steps rather than producing a major new product.

This model is used for low to medium risk project. When consideration of risk, funding, schedule, size of the program, complexity of program or need for early realization put into consideration then this model is used.

[19] http://www.qualitytesting.info/profiles/blogs/sdlc-incremental-model

## Agile Software Development[20]

Agile software development is a set of software development methods, which is based on iterative and incremental development and prototype. Here requirements and solutions develop based on collaboration between self-organizing and cross functional teams. It supports adaptive planning, evolutionary development and delivery, iterative approach, and promotes speedy and flexible response to change. It is a conceptual framework that supports the requirement of stakeholder throughout the life cycle of the software development.

Agile methods break up the tasks into small portions with minimum planning. It does not include long term planning. First working software is delivered in short time frame by using iteration approach. Normally, during iteration many activities will occur in parallel, such as requirements, coding and testing. Iterations are of a fixed length and thus are referred to as time-boxed. The time allocated to iteration is occasionally referred as a cycle time also.

Iterations involve a team, working through full software development cycle, including planning, requirement analysis, design, coding, unit testing and acceptance testing. First working software is gone through this developing stage.

This minimizes overall risk of project failure and allows the project to adapt to changes quickly. Team members produce documentation as required. First iteration might not add enough functionality to first working software. But multiple iterations may be required to release a product or new features.

Team members of agile project are usually organized by themselves. They take responsibility for tasks that deliver the working software with desired functionality iteration requires. They decide individually how to meet iteration's requirements.

Agile methods give attention to face to face communication over written documents when the team is all in the same location. When a team works in different locations, they maintain daily contact through videoconferencing,

[20] http://en.wikipedia.org/wiki/Agile_software_development

voice, e-mail etc. Team size is small about having 5-9 people to simplify team communication and team collaboration.

Each agile team contains a customer representative in their team. This person is appointed by stakeholders to take steps on their behalf and makes assurance to being there for developer team members to answer mid-iteration problem-domain questions. At the end of iteration, stakeholders and the customer representative review the progress and re-evaluate priorities with a view to get maximum return on investment and ensuring development process with customer needs and company goals.

Agile implementations use a routine and formal daily face-to face communication among team members. This particularly includes the customer representative and any interested stakeholders as observers. Team members pass on information to each other what they did the previous day, what they intend to do today, and what their problems are. This daily and face-to-face contact exposes problems as they arise. This type of meeting regularly held every day and should be completed within 15 minutes.

Agile development stress on working software as this is the primary measure of progress. This method strongly support on software development. This method does not contain require much of written documents. Agile method encourages stakeholders to put forward their requirement for next set of iteration.

There are twelve principles on which Agile Model is based. Its principles are defined in its declaration, Manifesto for Agile Software Development. One of the most main characteristics of this manifesto is that it is worded as a value statement and not a concrete plan or process. This defines that a development basis on Agile is with the values described in the manifesto.

### *Manifesto of Agile Software Development*

1) Customer satisfaction by rapid delivery of useful software

In this model, customer will not only given documentation of their requirements and about plan, how to develop the software? But within short

period of the software development process, customer will be given first working software.

2) Welcome changing requirements, even late in development

Stakeholders are encouraged to reveal their needs or requirements, which can accommodate in forthcoming iteration. So any time customer can put forward their requirements.

3) Working software is delivered frequently within weeks rather than months

Once the process start, the working software is delivered to the customer within the gap of some months. Consecutive later version will have more facility and more nearer to the required system or software.

4) Working software is the principal measure of progress

First working software is delivered early and within short time span of life cycle model. By using this model, stakeholder or user can get the actual feel of the software. After using this software, they can find another requirements and needs or facilities to be included within that software. After completing iteration, working software is delivered to the customer, which shows the how much the latest software is nearer to the actual desired software.

5) Sustainable development, able to maintain a constant pace

The changes are welcome at any stage of development; team has to regular with their work and gives response to the customer against newly arrived requirements.

6) Close, daily co-operation between business people and developers

One of the people from stakeholder side is committed to the developers to express their requirements to the developers. So, daily communication is there between business people and developers.

7) Face-to-face conversation is the best form of communication (co-location)

Generally agile methodology is suitable when the working team is at the same location. If they are not at the same location then they should be communicate with each other by video conferencing. To avoid problem to be arise at later stage, daily communication must be there within team members.

8) Projects are built around motivated individuals, who should be trusted

As there is no interference from managerial staff, all the team members must be trustworthy people. They are working at the same level so they must have to motivate themselves or motivate each other.

9) Continuous attention to technical excellence and good design

The working software is iteratively delivered to the customer so customer can find any problem within the software at the same time. If they are not comfortable with any sort of thing related to the software, they can ask for change to the developer. So developers have to pay continuous attention to the software under construction.

10) Simplicity- The art of maximizing the amount of work not done - is essential

Here developing is done at very fast and constant speed. So, all the developer must have to be alert to complete the task as early as possible.

11) Self-organizing teams

The members involve in the agile development are self organized. They all are work at the same level. No interference of the managerial side during the time of development. They are committed to their work.

12) Regular adaptation to changing circumstances

Working software is delivered to the customer iteratively and customer demanding changes continuously. So, team member must develop software in such a way that software can adapt these changes.

Agile methods differ from each other in the activities and work products that they produce. All the agile methodologies are not as accurate as traditional methodologies, but in current scenario these methodologies are much used by developers.

There are some Agile Development Methodologies like:

1) Rapid Application Development
2) Joint Application Development

3) Scrum

4) Extreme Programming (XP)

These methodologies are described below.

### 2.4.7 Rapid Application Development (RAD)

Today's world requires much faster ways of doing things. Machines hard disks spin faster, networks transport bits faster, and the real bottleneck seems to be creating software faster. Team members are also trying out much of new techniques to get new products into customer's hands quickly. We are limited by how fast we can code. Because the developers or programmers are passionate with knowing every line of code that was ever written, this will diminish the speed of programmer. Rapid Application Development is the solution of this problem. Starting with the ideas of Barry Boehm and Scott Shultz, James Martin developed the Rapid Application Development approach during the 1980s at IBM and finally formalized it by publishing a book in 1991[21].

"The fundamental principal of RAD is to remove fixed milestones from the project by delaying the production of detailed system design documentation and deliverables until user requirements are fully clear. This is done using prototyping". Szekely identifies prototyping as "a small scale version of a complex system in order to acquire critical knowledge required to build a system". Constant change in the prototype by the user and the developer results in a more tangible system development project that can respond to user requirements more quickly and easily. In other words, having prototyping concept, RAD can better deal with real life issues and so, user requirements are satisfied to a higher degree than within a SDLC framework.

Rapid Application Development is a software development methodology that needs minimal planning for development. The planning of software developed using RAD is interleaved with writing the software itself. The lack of much pre-planning is allowed here in RAD and software is written much faster, and makes it easier to change requirements.

[21] http://www.selectbs.com/analysis-and-design/what-is-rapid-application-development

Rapid Application Development gives innovative tools. To develop the system fast enough, one need tools to analyze, design and build system quickly.

RAD tools offer a set of reusable components that can be readily used in a solution; productivity could be improved and also decrease development time of developer. One of the big advantages of using such component frameworks is that testing could be done as component is build, which facilitate developer. It is perfectly true that code cannot be completely authenticated if it is not fully functional. But by using the component approach, each block is fully functional by itself and making it easy to test a component as it is built. Developer team can decrease development time by stating with working code or working component. Suppose like, a form component is added, and then some records are added or adding a new component like a report component and verifying what it generates. Developer could add new components to the already developed component and try out those components easily and end up with a complete and complete validated solution with the end of a few iterations.

Structured techniques and prototyping of RAD model are used to classify users' requirements and to design the final system. The development process initiates with the development of preliminary data model and business process model using structured techniques. In the next stage, requirements are verified using prototyping, to improve the data and process models. These stages are repeated iteratively, that results in new system to be ready for use.

RAD approaches may compromises in functionality and performance of the software which is developed in exchange for enabling faster development and facilitating application maintenance.

### *2.4.7.1 Core Elements of Rapid Application Development*

RAD has many core elements that make it a unique methodology including prototyping, iterative development, time boxing, team members, management approach, and RAD tools.

**1) Prototyping**

The main goal of RAD is the construction of a prototype for the purpose of early starting design and flushing out user requirements. The main aim is to build a light version of the finished software in a very short time. The initial product serves as tool for improving requirements. Developing prototypes so early is done with Computer Aided Software Engineering CASE tools that focus on capturing requirements, converting them to a data model, converting the data model to a database, and generating code all in one tool.

**2) Iterative Development**

Iterative development means creating increasingly functional versions of a system in short development cycles.

Each developed version is analyzed by the client to suggest more refined requirements that will be input of next version. The process is continuously repeated until all functionality has been developed. The best length of iterations is about one day to three weeks. Each development cycle gives the user a chance to give feedback, refine requirements, and view progress.

**3) Time Boxing**

Time boxing is the process of setting off all the features to future application versions in order to complete the current version in as short amount of time as possible. It is important tool that within this decided time line the iteration should be completed. This is an important tool of RAD because without it iteration might be lengthy.

**4) Team members**

This methodology advises the use of small teams. The team members should be experienced, adaptable, and motivated members that are able to perform multiple roles. User plays a very important role in the development process. Dedicated client resources must be available during the initial sessions as well as having focus on the development process. Development teams should have experience in Rapid Application Development and with the Computer Aided Software Engineering Tools.

### 5) Management Approach

Active and involved management is important to decrease the risks of development cycles, client misunderstanding, and missed deadlines. Management must be firm, dominant and steady in their requirement to use the Rapid Application Development methodology. Taking care of strict timeline is not only enough but also management must focus on team member selection, team motivation and involve with the solving of technical and social issue.

### 6) RAD Tools

One of the primary objectives of the Rapid Application Development methodology is to take full advantage of the latest technology available to speed up the software development.

There are so many tools available for Rapid Application Development. They are like requirements gathering tools, data modeling tools, code generation tools, data integration, development environments.

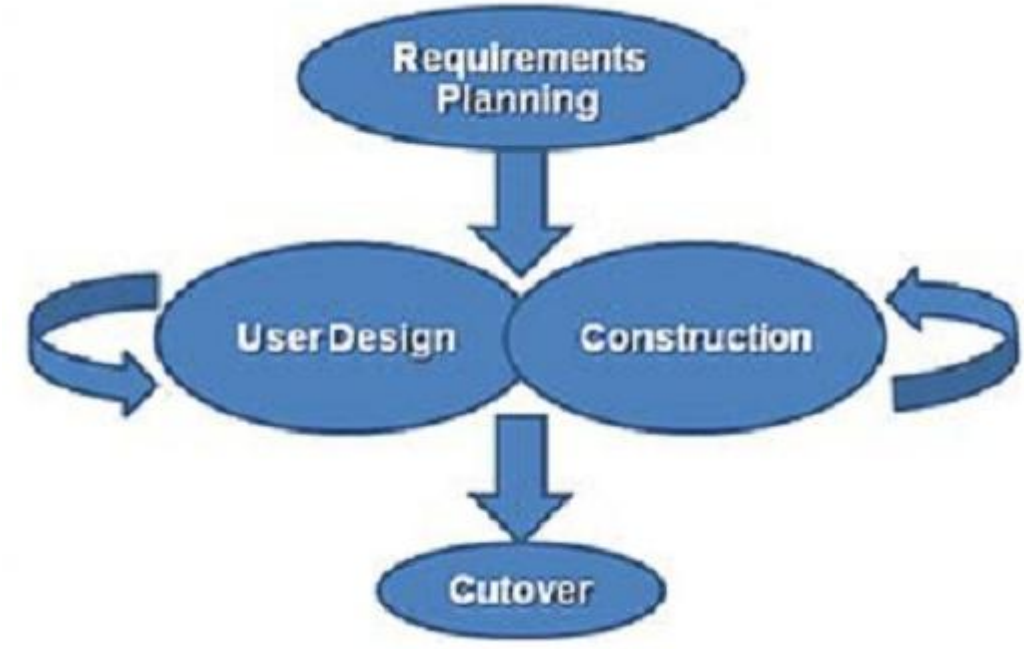



**FIGURE-13 RAPID APPLICATION DEVELOPMENT MODEL**

### *2.4.7.2 Phases of Rapid Application Development model*[22]

**1) Requirements planning phase**

It combines elements of the system planning and systems analysis phases of the System Development Life Cycle (SDLC). Users, managers, and IT staff members discuss and agree on business needs, project scope, constraints, and system requirements. It ends when the team agrees on the key issues and obtains management permission to continue with the next phase.

**2) User design phase**

In this phase, users communicate with system analysts and develop models and prototypes that represent all system processes, inputs, and outputs. The RAD groups or subgroups typically use a combination of Joint Application Development (JAD) techniques and CASE tools to convert user needs into working models. User Design is a continuous interactive process that allows users to understand, modify, and eventually approve a working model of the system that meets their needs.

**3) Construction phase**

In this phase, Focus is on program and application development task which is similar to the traditional SDLC. In RAD, users continue to participate and can still recommend changes or improvements as actual screens or reports are developed. Its tasks are programming and application development, coding, unit-integration and system testing.

**4) Cutover phase**

In this phase, the final tasks as in the SDLC implementation phase, including data conversion, testing, is done and use can changeover to the new system, and user training is there in this phase.

Compare with traditional methods, the entire process is compacted. As a result, the new system is built, delivered, and placed in operation much sooner. Its tasks are data conversion, full-scale testing, system changeover, user training.

[22] http://en.wikipedia.org/wiki/Rapid_application_development

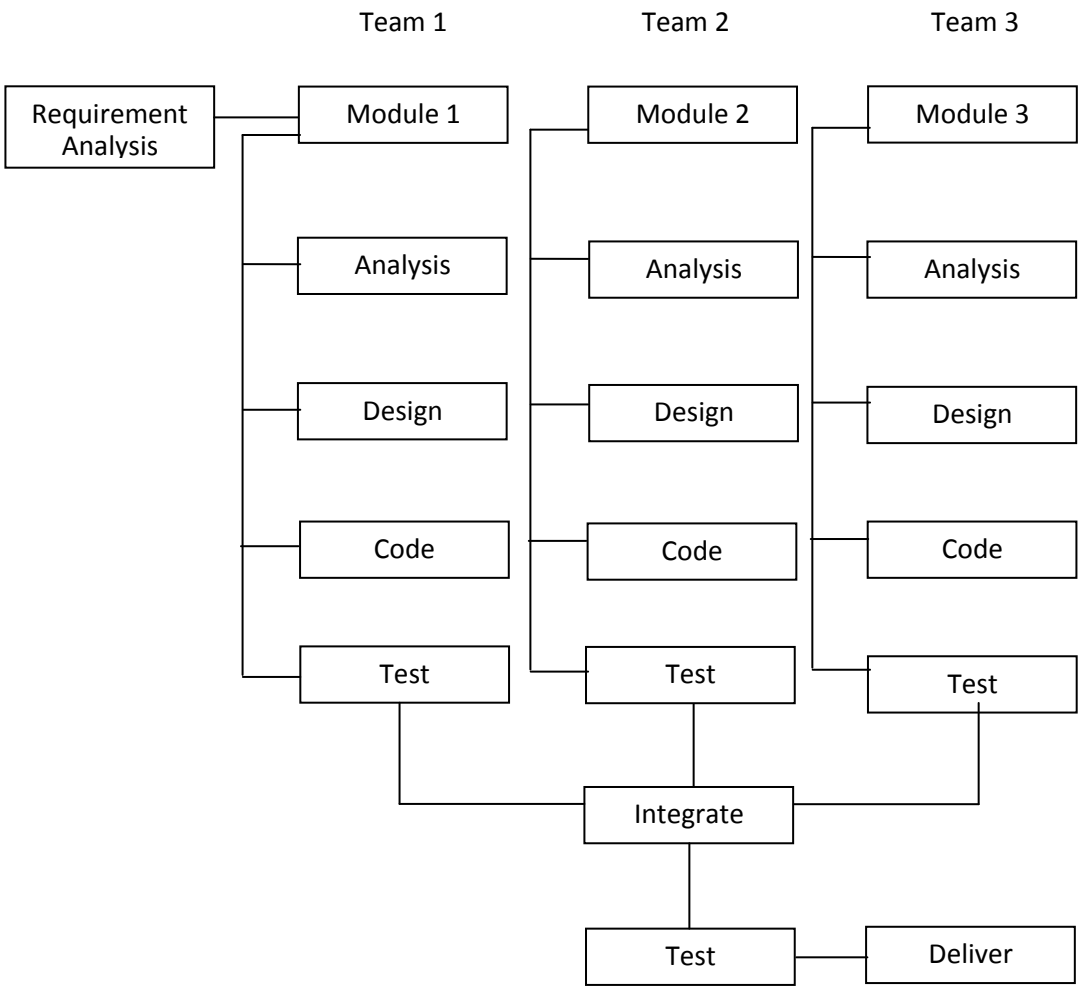



**FIGURE-14 RAPID APPLICATION DEVELOPMENT**

### *2.4.7.3 Advantages of the Rapid Application Development*

1) Working software is available much earlier than any conventional method.
2) RAD produces systems more quickly and to a business focus, this approach tends to produce systems at a lower cost.
3) A greater level of commitment is there from stakeholder. So user feels as gaining more of sense of ownership of a system, while developers feel gaining more satisfaction from producing successful systems quickly.
4) Focus on very important system elements from user viewpoint.

5) Provides the ability to rapidly change system design as demanded by users.
6) Gives strong connection between user requirements and system specifications.
7) Saving time, money and human efforts by rapid development.
8) Changing or stopping the way of development on a product that is not meeting its objectives is not harmful in terms of cost and effort.
9) Resulting final product often match user's needs and expectations very closely.

### *2.4.7.4 Limitations of the Rapid Application Development*

1) More speed and lower cost may lead be developed a poor system quality.
2) Due to inappropriate information, developed system might be misaligning.
3) Project may end up with more resources than needed.
4) Due to hurry, there may inconsistent designs within and across system.
5) There may be violation of programming standards related to inconsistent naming conventions and inconsistent documentations.
6) There can be lack of scalability in design.
7) Difficulty with module reuse for future systems.
8) High cost of commitment on the part of main user personnel is there as one has to remain present throughout whole development.
9) Formal reviews and audits are more difficult to implement than for a complete system.
10) Tendency for difficult problems to be pushed to the future to demonstrate early success to management.
11) RAD prototyping can be difficult to manage in large organizations.
12) User can be misleading to adopt premature working prototype as the finished product.

### *2.4.7.5 Where to use Rapid Application Development*

This methodology is used when user requirements are not certain and user's vision is not clear about the system.

Rapid Application Development is not appropriate for all projects. The methodology works best for projects where the scope is small or work can be broken down into manageable units. Along with this, project teams must also be small, preferably two to six people, and the team must have experience with all technologies that are to be used.

RAD is also recommended when system components have already been developed by the organization in the context of other software systems, and these components can be used in the new system with minor or no modification

RAD helps to reduce risk by pulling the business users in more tightly into the lifecycle. RAD works best with JAD.

RAD was a step toward adaptive approach; it did not have enough specific tools and techniques to deal with a constant flow of changes coming from the business to allow functioning effectively.

The RAD model is used when needs and solutions can be divided into modules as independent system or software components, each of which can be developed by different team. After then they are combine to produce the larger system solution.

RAD approach is good when a module of a larger system can be clearly defined in scope, functions, data, its processes, applications and outputs. The RAD approach is efficient when system modules are of same type in design, architecture and technology and there would not have any problems in smooth integration.

### 2.4.8 Joint Application Development (JAD)

JAD is requirements definition and user-interface design methodology in which end-users, executives, and developers attend powerful off-site meetings to work out a system's details. In this methodology client and developers together work out on design and client completely involve in development of an application. That's why this is known as joint application development. It is developed by Tony Crawford and Chuck Morris at IBM[23].

This is achieved through a series of combined workshops called JAD sessions. Two employees of IBM, Chuck Morris and Tony Crawford developed the JAD methodology in the late 1970s.

By the late 1980s, many companies were implementing facilitated JAD workshops for analysis and design. Because JAD has evolved over the years to include such elements as prototyping, CASE, some people consider it a complete development methodology and have begun to call it "joint application development." Unfortunately, the only portions of a generic development methodology for JAD that were formalized were the definition, analysis, and design portions[24].

In this methodology, a team is made up of members of different backgrounds. These members are made up of end-users, management, and IT staff. This team meets and discusses the current project in a number of workshops sessions. These workshops are used to describe the whole project and its requirements, scope, cost etc. and to design a solution. Since the team includes actual users of the current system, the system analyst will have a better picture of what the clients need or want in a new proposed system.

The prime focus of JAD is on the business problem rather than technical details. It is applicable to the development of business system, but it can be used successfully for system software. Time is saved by decrease the elapsed time required to gather a system's requirements and by gathering requirements better. This will save the cost of development as requirement is quite clear in the starting of the development so, later changes in the

[23] It is cited that in late 1970's, it is developed at IBM.
[24] http://www.umsl.edu/~sauterv/analysis/488_f01_papers/rottman.htm

requirement could be avoided. Success of workshops depends on JAD sessions. Participants of these sessions would normally include a facilitator, end users, developers, observers, mediators and experts.

There are multiple JAD participants. The first member is the sponsor. This is the executive of the organization who letting the project to be done. The second members are the business users. These users are the actual end-users of the system. The end-users will give detailed information on the day-to-day operations of the system. The next group of members is the IT professional. These members can be the inside IT staff and/or outside consultants. The last participants are the record keeper. This person documents everything that happens in these JAD sessions.

There are four thoughts to consider when working with the JAD session. First is, the business professional can understand the job function of their field very clearly so, they can share the best information of the business and their need during session. While second is, IT people can understand the current working system much better as anyone can do. They can answer any of the questions about this topic. These two points introduce the third idea, which is the difference between business professionals and IT professionals. At times, business professionals do not understand subjects in the IT area as they are "computer illiterate”. On the other side of the, IT professionals have issues understanding subject matter in the business area as they are "business illiterate". The hard task is to connect that communication gap. This gives the fourth idea. This idea is to grant all parties in the JAD equal partnership in the project. Everyone in the group needs to be equal.

JAD allows for a faster development process and minimizes errors at the same time. JAD also improves the quality of the final product by focusing on the up-front portion of the development lifecycle, thus reducing the finding of errors that are expensive to correct later on the way of development process.

JAD is thought to lead to shorter development times and greater client satisfaction, in which client is constantly involved.

The base of JAD is 'JAD sessions' which are meetings between the customer team and the developers where the specification is considered. Different

expressions of the JAD method vary slightly but the core components of a JAD session can be defined in terms of the attendants and their roles at the meeting as follows.

### *2.4.8.1 JAD Sessions Attendants*

**The Facilitator**

The facilitator arranges the meetings which is challenging the challenging task. The chances of expanding the time of meeting would be increase as if there is any problem with understanding the problem. JAD sessions are quite lengthy and tough. The facilitator is in charge for recording notes regarding issues, especially issues closed and opened. Usually, the facilitator is not engaged in the analysis process himself but has to manage the process of discussion.

The facilitator is the main person in the team and he is the ultimately responsible for the planning, execution and managing the project. To choose a facilitator is the initial significant step. He or she should be a respected. He /she must be a skilled leader with a good status within the organization. The skill to handle the session within the facilitator does not happen by chance and the skills may have to be learnt. JAD experience is also necessary. To choose a poor facilitator could lead to poor project development. So to choose a poor facilitator will result into poor outcome of the whole process. It is necessary that the facilitator is given the rights as well as the responsibility and will work personally with the Management sponsor to accomplish the objectives of the JAD project. The facilitator will know how to handle people, to be able to get the best out of them.

**The Sponsor**

For any computer project to get success, it will need the support of management. It is very essential for the JAD team to have a sponsor. The person may be a divisional head or manager of the business area that the JAD project is running. The sponsor does not have to actively participate in every JAD session. It might be appropriate to attend the first and may be the

final JAD session to check the results and make comments. The sponsor should be available throughout the period of the whole JAD development to answer any serious problems or issues that might arise during the development period. The JAD facilitator would work personally with the management sponsor and be kept fully aware on progress of the development. The sponsor can be from the end user community.

**The Project manager:**

Typically the project manager resolves issues regarding scope, resources and organizational structure. Normally, they are not part of the analysis team.

**Modeller:**

Modeller keeps records on design decisions and builds the formal model using whatever format the project is using. The Modeller has to take out each meeting's design decisions, revises the core design model, and also make available to the whole team and stakeholders. Normally, the modeller contributes in clarify the things.

The modeller has a significant role in the JAD team. He or she is responsible for documenting the JAD sessions. This must be done in an interactive manner and the modeller must work closely with the JAD facilitator. Many ideas and suggestions will be discussed between modeller and facilitator. The modeller must have to learn to capture the important decisions made, who made them and why. This information is a record of what was discussed in the meeting. Because of the skill need to process this information laptop computers are particularly useful as they are portable.

During JAD sessions it is needed to encourage end users to call on the modeller and converse with them to make sure about the points are documented. The modeller requires a logical approach to handle documentation. It is the duty of the modeller to share out the documentation to each participant at the end of each JAD session. It is a difficult task as well as very vital task and that cannot be underestimated.

**End users:**

These are customers, in the business area affected by the project, who are experts in their field and can make decisions about their work.

The role of end users is in designing the system and agreeing the systems. They make sure that the proposed system would be accepted as essential to its ultimate success. Without end user involvement JAD will not get success. The main purpose of JAD is to bring end users and Data Processing people together in a structured environment.

End users quickly take involvement and ownership in systems where JAD is used. This is a vital to its overall success. Most important, end users get the system they want which suit their need and requirement, not one which has been designed so poorly.

End users will invest the time and effort when they can see positive results.

**Information specialists or Analysts**

Information specialists who ask questions to the participants to make clear a formal picture of what processes are to be supported to develop the proposed system. Other things are also formalized like requirements, goals and deliverables.

Information specialists are there to help the end users in defining their requirements and build up a design according to the end users' requirements. Under the direction of the JAD facilitator, after discussing the requirements within the JAD session information specialists will need to create prototypes. So, complete knowledge is required to build prototyping software.

They can also recommend end users for new technology or hardware that can be use in the technical implementation. They need to understand the organization and the business area both are involved. Information specialists should be good listeners and would be able to understand end users. Experienced systems analysts who can use software have been excellent in this role.

**Observers:**

Observers are any customer or developer with an interest in the project. They have to sit physically outside the group. They should not involve in the discussion and says nothing during the meeting.

### *2.4.8.2 Pre-Workshop Activities*

Preparations for to arrange the workshop are be made to make them successful. There is almost one to three weeks preparation needed for a workshop. That preparation is required to:

1) Identify project objectives and limitations

It is important to have clear goals for the workshop and for the whole project. The pre-workshop activities, the planning and scoping, set the criteria of the workshop sponsors and participants. Scope of the project will identify the business functions that are within the scope of the project. It also tries to assess both the project design and implementation complexity. How many false starts were there? How many implementation failures were there? All the answers of such questions should be analyzed to decide the milestone or criteria of the project.

No of the workshop being held is also important. For best results, systems projects should be sized so that a complete design - right down to screens and menus - can be designed in 8 to 10 workshop days.

2) Identify critical success factors

It is important to identify the critical success factors for both the development project and the business function being studied. How will we know that the planned changes have been effective? How will success be measured? Planning for outcomes assessment helps us to judge the effectiveness and the quality of the implemented system over its entire operational life.

So, it must be identify the factors which could lead the proposed system to the success and meets the need of the end user.

3) Define project deliverables

In general, the deliverables from a workshop are documentation and a design. It is important to define the form and level of detail of the workshop documentation. What types of diagrams will be provided? What type or form of description will be supplied? The output of the workshop in form of deliverable should be already defined.

It is a good idea to start using a CASE tool[25] for diagramming support right from the start. Most of the available tools have great diagramming capabilities but their descriptive support is generally weak, so the descriptive support should be got from the standard word processing software.

4) Define the schedule of workshop activities

Workshops vary in length from one to five days. The initial workshop for a project should not be less than three days. It takes the participants most of the first day to get comfortable with their roles, with each other, and with the environment. The second day is spent in learning to understand each other and developing a common language with which to communicate issues and concerns. By the third day, everyone is working together on the problem and real productivity is achieved.

After the initial workshop, the team-building has been done. Shorter workshops can be scheduled for success of the project. However, it will take the participants from one to three hours to re-create the team psychology of the initial workshop.

5) Select the participants

These are the business users, the IT professionals, and the outside experts that will be needed for a successful workshop.

6) Prepare the workshop material

Before the workshop, the project manager and the facilitator make an analysis and build a preliminary design to focus the workshop. The workshop material made of documentation, worksheets, diagrams, and with the other physical

[25] They automate project management activities; manage all work products produced throughout the process, and assist engineers in their analysis, design, coding and testing work. They are like UML model, use case model, etc.

thing that will help the participants understand the business function under analysis.

7) Organize workshop activities and exercises

The facilitator must design workshop exercises and activities to provide temporary deliverables that build towards the final output of the workshop. The pre-workshop activities help design those workshop exercises. For example, for a Business Area Analysis, a de-composition diagram, a high-level entity-relationship diagram, a normalized data model, a state transition diagram, a dependency diagram. To get such formalized information workshop activities must be organized in such a way that all the related matter must be discussed in a linear format. It does not create messy environment. Technical diagram should be prepared in such a way that it must be understood by the users.

Once the diagram choice is made, the facilitator designs exercises into the workshop agenda to get the group to develop those diagrams. A workshop combines exercises that are successively oriented to build on one another, and parallel exercises, with each sub-team working on a piece of the problem or working on the same thing for a different functional area. Some exercises led by the facilitator energize the group and direct it towards a specific goal. Some exercises allow for detailed discussions before decisions. To discuss about the solution of the problem total group or team members are included and they are allowed to present a limited number of suggestions for the whole group to consider.

To incorporate with the participants, the facilitator can look for people with similar expertise from different departments and match them at the workshop. To help participants learn from each other, he can mix the expertise. It's the duty of the facilitator to mix and match the sub-team members to complete the organizational, cultural, and political objectives of the workshop.

A workshop operates on both the technical level and the political level. It is the facilitator's job to build harmony and communications, so that participants can discuss about the project. There is no need to worry about the technical

implementation of a system if the underlying business issues cannot be resolved.

8) Prepare, inform, and educate the workshop participants

All of the participants in the workshop must be made aware of the objectives and limitations of the project and the expected deliverables of the workshop. Updating the participants should take place 1 to 5 days before the workshop. This updating may be teleconferenced if participants are very far away from the location.

The updating document might be called the Familiarization Guide, Briefing Guide, Project Scope Definition, or the Management Definition Guide - or anything else that seems appropriate. It is a document of eight to twelve pages, and it gives a clear definition of the scope of the project for the participants.

The updating itself takes almost to complete two to four hours. It provides the psychological preparation for everyone who needs to move forward into the workshop.

9) Coordinate workshop props

Workshops should be held off-site to avoid interruptions. Projectors, screens, PCS, tables, markers, masking tape, Post-It notes, and lots of other props should be prepared.

What specific facilities and props are needed is up to the facilitator. They can vary from simple flip charts to electronic white boards. In any case, the layout of the room must promote the communication and interaction of the participants.

It is know that human performance for solving a task is directly related to how a task is understood by the people attempting to solve it. If concentration is made on techniques to increase the understanding it shall increase developer performance. Also performance is influenced by personality and intelligence. We can do something about intelligence by providing training. Changing a personality is much more difficult.

When JAD is introduced motivation and performance improve dramatically. Because JAD focuses on analyzing the task and providing a solution, it has benefits in performance and motivation of the individuals taking part.

### *2.4.8.3 Advantages of the Joint Application Development*

1) Improve the communication between business users and the project team.
2) Enhances quality of the software.
3) It allows for the simultaneous gathering and consolidating of large amounts of information.
4) The experts get a chance to share their views, understand views of others, and develop the sense of project ownership.
5) Creates a design from the customer's perspective. It means customer can influence in software design.
6) It reduces organizational infighting.

   By bringing all the decision makers together to design the system, JAD brings conflicting objectives and hidden agendas to light early in the project, when they can still be addressed effectively and before they've had time to do much damage.

7) Project teams get focused and stay focused.

   In the workshop, the participants will build a common view of the project and a common language to discuss the issues. These elements will stay with the team for the life of the project.

8) A natural partnership with modern development tools.

   JAD helps in realize the full potential of today's powerful development tools by providing high-quality input requirements quickly.

9) Improves design quality.

   The JAD improves the quality of the deliverable of the design phase because it forces a definition of that deliverable in advance. During the

workshop the participants are all focused on a common goal. Users in the workshop will have a better understanding of the business issues, the systems issues, and the volume of work to be done.

10) Enhanced with updated information for participants and observers.

By participating in JAD users and IT professionals, the business end-users will be kept fully informed about the progress of the system development.

11) Enables rapid development.

In JAD, information can be obtained and validated in a shorter time frame by involving all participants who have a stake in the outcome of the session. As the design approves, the development would start.

12) Improved quality of deliverables

Much of the system's quality depends on the requirements gathered. JAD involves users in the development life cycle, lets users define their requirements, and thus ensures that the system developed satisfies the actual activities of the business.

13) Reduced system cost.

Much of the system development cost is in terms of man-hours of both system developers and business users involved. Reduced development time reduces the labour cost for developers, as well as users. JAD can reduce the involvement time of business experts and hence reduce the cost further. Cost is also reduced by catching errors, misunderstandings and mistakes early in the development phrase.

### *2.4.8.4 Limitations of the Joint Application Development*

1) JAD is more expensive and can be cumbersome if Project is large.

If the group is too large relative to the size of the project then it would be very difficult to manage that group in JAD workshop.

2) Highly expert people are required to manage JAD Project.
3) User can take domination for the project development.
4) Continuous changes might delay the project development.
5) Time commitment

Depending on the size of the project, a JAD may require a significant time commitment. All JAD participants must be able to meet at the designated times and will need to suspend all other activities for these periods.

### *2.4.8.5 Where to Use the Joint Application Development*

JAD is usually used when project time span is 3 to 6 months long. It is used for low to medium scale project and the JAD uses the format to develop the project is incremental project.

When one of the stakeholders can constantly gives the time to the development team throughout development process at that time this methodology is used.

JAD is a user oriented technique for fact finding and requirements modelling. JAD can be used whenever group input and interaction are desired. JAD typically focuses on fact finding and requirements determination.

Cost of development is very moderate. Due to this, feature when requirement of the system is immediate at moderate cost with average no of team then this model is used.

### 2.4.9 Scrum

In rugby, scrum means mass of good players engaged with each other to get a job done. In software development, the job is to put out a release. Scrum for software development came out of the rapid prototyping community because developer wanted a methodology that would support an environment in which the requirements were not only incomplete at the start, but also could change rapidly during development. Scrum methodology includes both managerial and development processes.

It is described by Hirotaka Takeuchi and Ikujiro Nonaka[26]. Scrum is an iterative and incremental approach for managing software projects and product or application development. Scrum focuses on project management institutions where it is difficult to plan ahead. Mechanism of experimental process control, where feedback loops that compose of the core management technique are used as opposed to traditional command and control oriented management. It represents a completely new approach for planning and managing projects, bringing decision making authority to the level of operation properties.

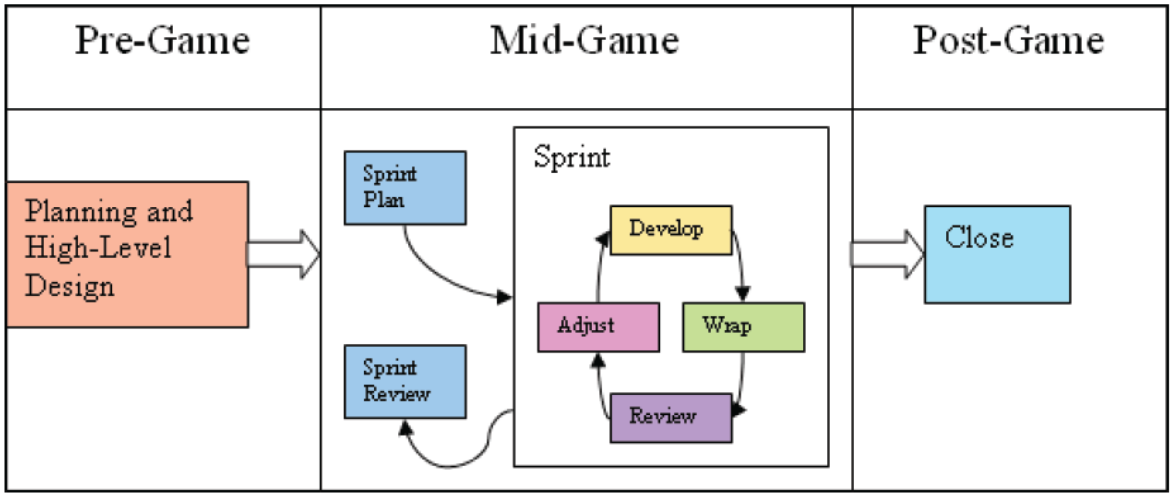



**FIGURE-15 SCRUM MODEL**

[26] In 1986 this approach is defined for commercial product development.

At the centre of each scrum project is a “backlog”, which is the defined list of work to be done. This backlog is written during the planning phase of pre-game and states the scope of the release.

After the team completes the project scope and high-level designs, it breaks up the development process into a series of small iterations called “sprints”. Each sprint has objective to implement a fixed number of backlog items. Before each sprint, the team members recognize the backlog items for the sprint. At the end of a sprint, the team reviews the sprint to check that the item which were selected are done and also checks the progress and get lessons for the next sprint are identified.

The Scrum development process focuses on how to manage sprints. Before each sprint starts, the team plans for the sprint, identified backlog items and assigning teams to those items to complete during the sprint. Teams develop, wrap, review, and adjust each of the backlog items.

During a sprint, the team has a regular meeting. Each member explains the work to be done that day, progress of yesterday work, and any problems that must be cleared. This meeting is short in time, the scrum meeting is supposed to be run with everyone in the same room.

During development, the team decides the changes necessary to implement a backlog item. The team then writes the code, tests it, and documents the changes. During wrap, the team creates the executable necessary to demonstrate the changes. In analysis, the team demonstrates the new features, adds new backlog items, and assesses risk. Finally, the team combines the data from the analysis to update the changes as necessary.

The sprint backlog is property of the development team. During a sprint; no one is allowed to edit the sprint backlog except the development team. The sprint goals should not be changed during the sprint. Development is timeboxed means the time limit is there to complete the sprint, such that the sprint must complete on time; if requirements are not completed for any reason then they are left out and that must have written back to the product backlog.

When enough of the backlog has been implemented so that the end users believe the release is worth putting into production, management put an end to development. The team then carries out integration testing, training and documentation as necessary for product release.

Following each sprint, the entire team—including management, users, and other interested parties—demonstrates progress from the sprint and reviews the backlog progress. The team then analyzes the remaining backlog and adds, removes, or reprioritizes items as necessary to account for new information and understanding gathered during the sprint.

### *2.4.9.1 Scrum Concept*

**1) Roles**

There are three main roles and other subsidiary roles.

**Core Roles:**

The core roles are those committed to the project in the Scrum process—they are the ones producing the product. They represent the scrum team.

**a) Product Owner**

The Product owner represents the stakeholder who is the voice of the customer. He or she is responsible for making sure that the team delivers value to the business. The Product owner writes customer-oriented items and prioritizes them, and adds them to the product backlog. Scrum teams should have one product owner, and while they may also be a member of the development team, it is necessary that this role must not be combined with that of Scrum Master.

**b) Development Team**

The Development Team is answerable for delivering required product increments at the end of each Sprint. A Development Team is made up of 3–9 people with cross-functional skills who do the actual work like analyze, design, develop, test, technical communication, document, etc. The Development Team in Scrum is self-organizing means task is not allotted to them but they are picking the task from the sprint.

### C) Scrum Master

Scrum is makes easy by a Scrum Master, who is liable for removing difficulty for the ability of the team to deliver the sprint goal or deliverables. The Scrum Master is not work as the team leader, but behaves as a safeguard between the team and any troubling elements. The scrum master ensures that the Scrum process is used as planned. The scrum master forces the team member to follow the rules. A main part of the scrum master's role is to protect the development team trapping into unnecessary issue and keep it focused on the tasks at hand. The role has also been known as a servant and leader to support these dual viewpoints.

### 2) Subsidiary Role[27]

The subsidiary roles in scrum teams are those with no formal role and not regularly involved in the scrum process. They must be taken into consideration.

### a) Stakeholders

The stakeholders are the customers, vendors. They are the people who enable the project and for whom the project produces benefit by selling it that justify its production. They are directly involved in the process during the sprint reviews.

### b) Managers

Managers are the people who can control the development environment.

**Meetings Required in Scrum:**

Daily Scrum: A daily sprint meeting

Each day during the each sprint a project status is calculated by meeting. This is known as daily scrum or the daily stand-up. This meeting has specific features.

- The meeting start in time.
- All members are allowed, but normally core members speak.
- The meeting time is set to 15 minutes.

[27] http://en.wikipedia.org/wiki/Scrum_(software_development)

- The meeting happens at the same time and same place every day.

In the meeting each member has to give report based on task allotted to them. They have to give answer to the following questions.

- What is the progress in task given to them since yesterday?
- What is planning for today?
- What are the problems or hurdles for task given to them?

If any problem is there, it is the task of scrum master to remove any problem or hurdles. The solution should come out outside the daily scrum meeting itself to keep it under 15 minutes.

**Backlog grooming or Story time**

The team should spend time during a sprint doing product backlog prepare. This is the process of estimating the existing backlog using point. Agree to the individual stories regarding the current product. The characteristics of the meeting are as follows.

- Meeting takes maximum of one hour.
- Meeting does not include breaking stories into task.
- The team decide how many meeting are needed per week.

Planning poker method is used to conduct such meeting.

**Scrum of Scrums:**

Each day after the daily scrum following activities are done.

- All the team members discuss their work, concentrating normally on areas of overlap and integration.
- A key person from each team attends.

That key person is responsible for the progress of his team. He or she can be asked following questions.

- What has your team done since last meet?
- What will the team do before the next meet?
- Is anything which slowing the performance of the team?

**Sprint Planning Meeting:**

At the beginning of the sprint cycle (every 7–30 days), a "Sprint planning meeting" is held.

- Identify the work to be done.
- Prepare the sprint backlog that details the work with the entire team.
- Identify and communicate how much amount of the work can be done during the current sprint.
- Eight hour time limit is given to team for prioritizing the product backlog and plan out to reach to the destination.

**Sprint Review Meeting:**

- Review the work that was completed and not completed
- Present the completed work to the stakeholders
- Incomplete work cannot be demonstrated
- Four-hour time limit

**Sprint Demonstration:**

- All team members reflect on the past sprint
- Make continuous process improvements
- Two main questions are asked in the sprint retrospective: What went well during the sprint? What could be improved in the next sprint?
- Three-hour time limit

### *2.4.9.2 Advantages of the Scrum*[28]

1) Scrum helps the company in saving time and money.
2) Scrum methodology enables projects where the business requirements documentation is hard to measure to develop successfully.
3) Fast moving developments can be fast coded and tested using this method.
4) Mistake can be easily rectified so development saves time.

[28] http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-agile-scrum-software-development.html

5) It is a lightly controlled method which insists on frequent updating of the progress in work through regular meetings. Thus there is clear visibility of the project development.
6) Like any other agile methodology, this is also iterative in nature. It requires continuous feedback from the user.
7) Due to short sprints and constant feedback, it becomes easier to cope with the changes.
8) Daily meetings make it possible to measure individual productivity. This leads to the improvement in the productivity of each of the team members.
9) Issues are identified well in advance through the daily meetings and so it can be resolved in speedily.
10) It is much easier to deliver a quality product in a scheduled time.
11) Scrum can work with any technology or programming language but is generally useful for fast moving web technology or new media projects.
12) The overhead cost in terms of process and management is minimal thus leading to a quicker result.

### *2.4.9.3 Limitations of the Scrum*[29]

1) There is not any specific end date, the project management stakeholders will be used to keep demanding new functionality to be delivered.
2) If a task is not well clear, approximation of the project costs and time will not be correct. In such a case, the task can be spread over several sprints.
3) If the team members are not committed, the project will either never complete or fail.
4) It is good for small, fast moving projects as it works well only with small team.

[29] http://www.my-project-management-expert.com/the-advantages-and-disadvantages-of-agile-scrum-software-development.html

5) This methodology needs experienced team members only. If the team consists of people who are new to this process, the project cannot be completed in time.
6) If scrum master is too strict to control over the team members, it can be very frustrating for them, leading to discouragement and the failure of the project.
7) If any of the team members leave during a development it can have a vast opposite effect on the project development.
8) Project quality management is hard to execute and measure if the test team is not able to conduct testing after each sprint.

### *2.4.9.4 Where to use the Scrum*

Scrum works better in environments with rapid changes. It is highly adaptable to the changing requirements. It is useful when fast feedback is given by stakeholders or users.

When team is passionate about its work or a manager will there to help them dedicatedly at that time scrum is used. Stakeholder is constantly in touch with the development team. The software which is being developed and wanted software exactly match.

When project is too complex then only Scrum is need. When iterative approach is there and requirements are not defined properly at that time scrum is well to be used.

Web based application can be developed using scrum model.

### 2.4.10 Extreme Programming (XP)

Extreme Programming is one of the well known agile methodologies. It is a programmer oriented methodology that based on technical practices to promote skilful development through frequent delivery of working software. This model is developed by Kent Beck[30].

Extreme Programming is successful because it stresses customer satisfaction. Instead of delivering everything you could possibly want on some date far in the future this process delivers the software you need as you need it. Extreme Programming empowers your developers to confidently respond to changing customer requirements, even late in the life cycle[31].

Extreme Programming emphasizes teamwork. Managers, customers, and developers are all equal partners in a collaborative team. Extreme Programming implements a simple, yet effective environment enabling teams to become highly productive. The team self-organizes around the problem to solve it as efficiently as possible[32].

One of the prime differences between XP and other methodologies is its cycle time and level of formal procedure. XP supports very small iterations between one and four weeks. XP is also a very low procedure methodology as it requires very small amount of artifacts. Minimal objects or artifacts[33], like story cards, code and unit tests are included in XP project.

XP is most commonly known for its technical practices. At the heart of XP are four core values: communication, simplicity, feedback, and courage. From these values 13 practices are derived.

#### *2.4.10.1 XP practices*

**1) Planning Game**

All the work is divided in small task. To complete that task planning is needed. Plan is made to work done incrementally. To work properly planning game

[30] XP is developed during 1996 by Kent Beck.
[31] http://www.extremeprogramming.org/
[32] http://www.extremeprogramming.org/
[33] This word introduced during the evolution of RUP. An artifact is a piece of information that is produced, modified or used by a process. Artifacts are tangible product of the project like source code, a class, an object, a design model, a document.

needs to be prepared. There are two types of planning game the release and iteration planning games. And each planning game is made by three phases: exploration, commitment, and steering.

Release planning starts with the customer writing story card and prioritizing them. Programmers then estimate the stories and from those stories a speed can be measured. Speed depicts how much time is required to complete the work. The customer chooses a range for release the product. This is decides at iteration boundaries when progress to the planned release date can be achieved so the adjustments can be easily made in the development.

Iteration planning follows a similar format that of release planning. Iteration also starts with developers taking stories and breaking them into tasks. Programmers take complete duty for tasks to complete and give estimation about the tasks. Each programmer's work load is compared with their previous work performance, to check if anyone is overcommitted and to allow the workload to be balanced across the team. During the whole iteration, programmer partners complete the tasks by writing unit tests and code. Throughout the whole iteration a member of the team continuously checks on the progress of the team and imparts this information to all team members so adjustments can be made.

**2) Small Releases**

As the working software is complete, that software is release as early as possible to the market. So, that the market time is increased. More people would use it and more feedback is collected which would help in next phase of iteration. The requirements or changes which are collected in the form of feedback are incorporate in next iteration.

**3) Metaphor or Simile**

If possible, give a metaphor for the system being developed. It is the name of system which having the same meaning for which purposes the system being developed. If proposed system developed is web application then it would be beneficial for customers to identify for which purpose the application is being used. For example, the shopping cart metaphor is widely used to describe an online ordering system.

### 4) Simple Design

Use the simplest design that will describe for the functionality or as described above “user story” being implemented. If the design is simple then it would be very easy to change according to the requirement. It must be taken care to design the system according to current requirements. Designs should not be made for the things that may never actually be used.

### 5) Testing

Testing, as defined in Extreme Programming, includes unit testing by programmers, functional testing by users, and automated testing to generate test units that test the actual programming units. This process totally supports the testing of programming units to encourage a fault free unit with each unit release. This type of testing makes the development process smooth. Testing is continuously done until no more tests can be uncovered. All modules are tested at the end of the iterations.

### 6) Refactoring

Instead of designing the entire system in one shot, designing is done as development progresses, make improvements as required. Change the implementation without changing the interface and its functionality, and use automated testing to determine the impact of the refactoring.

“Refactoring is an activity of constant re-designs of a program unit to take advantage of programming techniques particularly object-oriented design and design patterns, to make the programs more reusable, simpler, and more efficient.” Refactoring can be done at various times throughout the development process. For example, if during the programming process one of the programming pair determines that two methods are doing the same functions but with different data, the programming pair would re-factor the code to take advantage of polymorphism.

### 7) Pair Programming

Pair programming is the practice of two programmers sitting with each other and participating in the development as of one programming unit. Both of them are working on the same programming unit, generally with each member

performing a task the other is not doing. Usually, the pair is divided into one person entering code or test cases while the other is analyzing and thinking. For example, they might discuss a method which is required in their programming unit and decide to implement the new method. While one member is typing the method, the other member might be thinking of mechanisms for testing that method during development.

**8) Collective Code Ownership**

Anyone in the team can make a change to any code at any time as the code which is developed, is team effort. The code is not personal to any particular one programmer. So both of them are combined owner of the code prepared by them. Any one of team can change the code. It is right of the programmer to change the code if it enhance the scalability of the software developed by them but surely that modification does not harm the program or does not leak the software's security.

**9) Continuous Integration**

Continuous integration is the concept of to integrate new code into currently existing code and after then use the testing techniques to test the developed code or module. This practice gives units of code that are repeatedly tested during development. Using continuous integration, a programmer integrates new code into existing code after the existing code has been completely tested. Therefore, after completely change is done and software is completely tested and everyone knows about the new functionality which the software is going to deliver at the time of the release. The entire code base is constantly being rebuilt, and retested in an automated fashion.

**10) Sustainable speed**

Ideally, team members do not need to work more than 40 hours per week to meet project deadlines. But if management takes decision in favor of consistent, predictable, repeatable delivery then team members have to do the same to achieve the goal for the iteration decided by management. The software must be in market by the given time which is already decided. So, the entire team member must have to work at average speed with good patience to cope with the goal.

### 11) Whole Team

The team works as a whole. The team members have to work together. Members are encouraged to be more generalized than specialized. Learning about all technologies and requirements is encouraged which facilitates the software development. Anything concerning to software must be known by all the members of team. Team members can experiment with the code to make the software more scalable. All the members work as a team not as individual.

### 12) Coding Standards

In order to maximize communication, coding standards are defined by the team, and used to ensure consistent coding practices. It is for that to make the software consistent. All the team members have to follow the standard defined by the management to maintain the consistency of the software.

### 13) Onsite Customer

Having constant and direct access to the customer allows the team to work at the fastest possible speed. It would be very easy for the developer to know the feedback of the customer if they are available all the time at site. System which is developed could meet the exact requirements of the customer by the continuous availability of the customer.

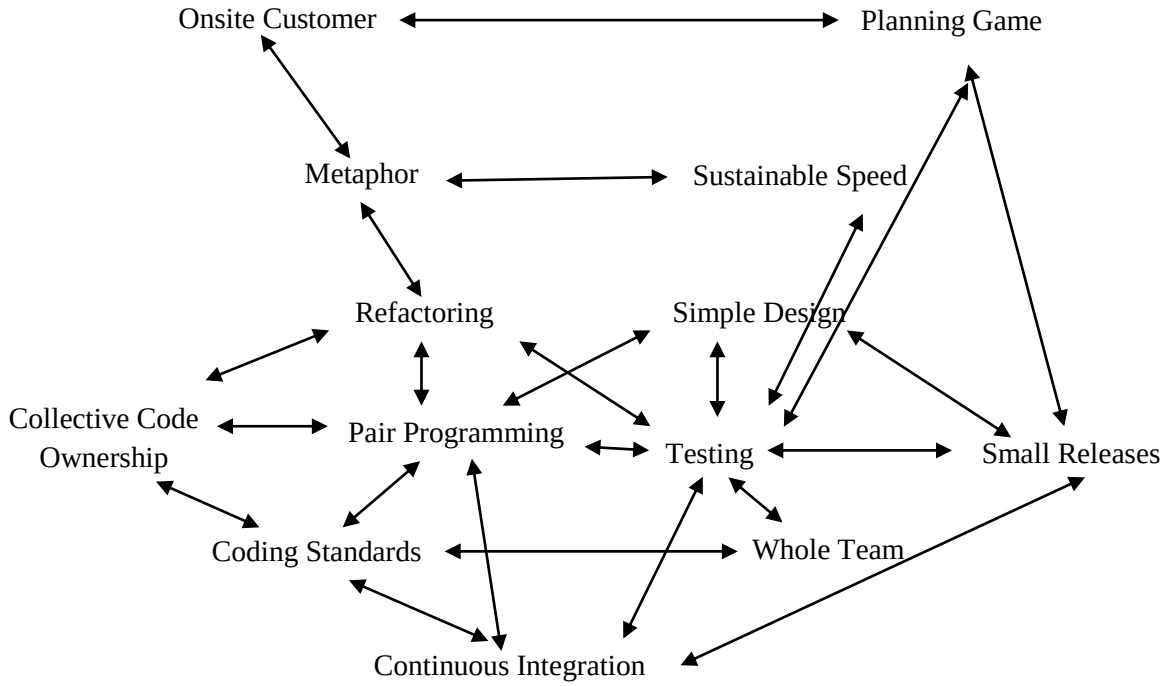



**FIGURE-16 PRACTICE RELATES TO ONE ANOTHER (XP)**

These practices support one another or we can say they depend on one another, as shown in above figure and thus care should be taken when modifying any of them, or deciding not to include one or more of these practices in a project.

The primary requirement object in XP is the user story. Actually, a user story is a note card with a short description on it. User stories actually are made of the card which has information about required functionality, conversations between developers and owner of the system that contains requirements, and tests.

There are a small number of roles on an XP project. These include:

- Customer

  The customer creates and prioritizes the stories which are the functionality which customer wanted to implement with the proposed software with respect to XP. Because the customer sets the priorities those will be delivered in any given release and can control the release date by adding and removing stories in terms of requirements.

- Programmer

  Programmers collectively estimate stories or functionality, but individually accept responsibility for tasks which they estimate, write tests, and code.

- Manager

  The manager monitors the software development process; guides team members on XP process and techniques, and concentrate the team's attention on potential problems.

- Tracker

  The tracker observes the progress of the team and alerts them when adjustments to the schedule or a re-allotment of tasks might be required.

XP describes four basic activities that are performed within the software development process: coding, testing, listening, and designing. Each of those activities is described below.

### *2.4.10.2 Basic Activities of XP*[34]

**Coding**

The followers of XP argue that the important product of the system development process is code, which are software instructions that a computer can interpret. Without code, there is no working product.

Coding can be used to give the most suitable solution. Coding can also help to communicate or represent the needs of the business by programming the problems. A programmer dealing with a complex programming problem or finding it hard to explain the solution then simplified code is developed.

Programming can be done in different way according to developer's personal practice. Other programmers can give feedback on this code by also coding their thought or according to their skill.

[34] http://en.wikipedia.org/wiki/Extreme_programming

### Testing

Extreme programming's approach is that if a little testing can eliminate a few problems, a lot of testing can eliminate many more problems.

- Unit tests determine if a given feature works as intended or not. A programmer writes as many automated tests as they can think of that might capture the loopholes in the code; if all tests run successfully, then the coding is complete. Every piece of code that is written is tested before moving on to another feature.
- Acceptance tests checks that the requirements as understood by the programmers satisfy the customer's actual needs.

### Listening

Programmers must understand about the customer’s requirement for the proposed system to do in the business or we can say what "business logic" is needed. They must understand these requirements well enough to reply for the customer feedback about the technical aspects of how the problem might be solved, or cannot be solved.

### Designing

It is very simple; of course one could say that system development is just group of activities like coding, testing and listening. If those activities are performed well, the result should always be a system that works. But in practice, this will not work. Developer can develop a system without designing but at that time developer will get stuck. The system becomes too complex and the dependencies within the system need to be clear for further development. One can avoid this by creating a design structure that classifies the logic in the system. Good design will avoid lots of dependencies within a system; this means that changing one part of the system will not affect other parts of the system.

### Values

Extreme Programming initially recognized four values in 1999. A new value was added in the second edition of Extreme Programming Explained. The five values are:

### Communication

To build software it requires communication about system requirements to the developers of the system. In formal software development methodologies, this task is completed through documentation. Extreme programming techniques can be seen as methods for quickly building and imparting such knowledge among members of a development team. The goal is to give all developers a shared view of the system which matches the view held by the users of the system. To this end, extreme programming supports simple designs, common metaphors, collaboration of users and programmers, regular verbal communication, and feedback.

### Simplicity

Extreme programming encourages starting with the simplest solution. Extra functionality can then be added later. The difference between this approach and more conventional system development methods is the focus on designing and coding for the needs of today instead of those of tomorrow, next week, or next month. Total focus would be on current requirements. Further enhancement can be done with the system being developed.

Supporters of XP acknowledge the disadvantage that this can sometimes need more effort to change the system tomorrow. Coding and designing for uncertain future requirements implies the risk of spending resources on something that might not be needed. Simplicity in design and coding should improve the quality of communication. A simple design with very simple code could be easily understood by most programmers in the team.

### Feedback

Within extreme programming, feedback relates to different aspects of the system development:

- Feedback from the system

  By running unit tests or running periodic integration tests, the programmers have direct feedback from the state of the system after implementing changes.

- Feedback from the customer

  The functional tests are written by the testers. They will get concrete feedback about the current state of their system. This review is planned once in every two or three weeks so the customer can easily push the development. So, the development can be done in the way which can fulfill the proposed requirements of the customer.

- Feedback from the team

  When customers come up with new requirements in the planning game the team directly gives an estimation of the time that will take to implement new requirements.

Feedback is directly related to communication and simplicity. The direct feedback from the system informs programmers to recode some part. A customer is able to test the system periodically according to the functional requirements as described in user stories.

**Courage**

Several practices represent courage. One is the demand to always design and code for today and not for tomorrow. This is an effort to avoid getting chocked in design and requiring a lot of effort to implement anything else. Courage facilitates developers to feel secure with refactoring their code when necessary. That means analyzing the current system and changing it so that future changes can be implemented easily.

No matter how much effort was used to create that source code, a programmer has to have courage to remove source code that is of no use. A programmer might be trapped with a complex problem for an entire day then can solve the problem quickly if he or she is continuously doing positive effort.

**Respect**

The respect value includes respect for others as well as self-respect. Programmers should never place such changes that break compilation, that make current unit-tests fail. Members should respect their own work by determined for high quality and finding for the best design at refactoring.

Adopting the four earlier values leads to respect gained from others in the team. Nobody in the team should feel unappreciated or ignored. This makes sure a high level of motivational and encourages loyalty toward the team and toward the goal of the project.

### *2.4.10.3 Advantages of the Extreme Programming*

1) Constant feedback from customer

   Developer gets constant and complete feedback from customer which helps them to add new features to the system which is being developed.

2) Customers are available onsite

   Customers are constantly available on site. So, when developer team require asking anything it would be very easy.

3) Pair programming gives better coding

   As programming is done in a team of two people, it would be beneficial to the coding. Because all the people in a team have different way of thinking relates to see a particular programming aspect. So better coding would be outcome of these aspects.

4) Refactoring is also beneficial

   Changes to the designing are admirable in XP. Instead of designing the entire system in starting, design as you go, making improvements as needed. So the interface and its functionality would not change.

5) Continuous Integration

   As new code is developed, it is tested and integrated with the old code. The entire code base is constantly being rebuilt, and retested in an automated fashion.

6) Silly mistakes are quickly caught

Simple mistakes such as syntax errors or repeated variable names can be easily caught and fixed right then and there. This might not seem like much, but it can cut down debug time later and prevent from small irritating bugs.

7) Better concentration

This isn't particularly scientific, but it seems that too people working have a lesser tendency to get distracted. This results in shorter development times too.

8) Its a good training ground for large software projects

Very little software is written by only one person, team size is varying from small to big are not just the standard, and this is the necessity. Pair programming teaches you a lot of the soft skills you'll need: tolerance, respect, understanding and responsibility.

9) Combining knowledge

Computer science is a vast field. It's hard for any single member to have a complete knowledge of what's been going on in software, so if two of them working together mean that they can share their knowledge and can work together.

10) Constant planning

The problem with most projects is they attempt to plan once and assume they have finished with it. They develop such mentality to work according to that plan only. They spend a lot of time initially to come up with a date that everyone knows it is of no use. XP takes the view that planning is good, but one must have to keep on planning and re-planning all the time. Reality always interrupts the plans.

11) It gives customers the ability to see whether or not a company can deliver on its promises.

12) It gives management many tools, including predictability, flexibility of resources, consistency, and visibility into what's really going on.

### *2.4.10.4 Limitations of Extreme Programming*

1) Skill disparity

   This is the number one potential problem. If the partners are of completely different skill levels, you might have one programmer doing all the work or constantly tutoring the other. This is ok if one wants to set up a teacher-student relationship or introducing a new programmer to your system, but if not, it can fail the entire purpose of XP.

2) Not actually getting the work done

   For some people pair programming concept can easily generate in social sessions. There are some people who can't work when there is someone next to them examining their work, these people will not benefit much either.

3) Developer egos

   This is something that is not likely to happen in a classroom, but in more experienced teams, each programmer might try to push their own ideas of how things should be done, both of which may be perfectly valid. These kinds of conflicts can be totally unsuccessful.

4) Could not release in time

   As members are working in teams, they might not be able to reach the target in time or complete the iteration in defined timebox. It might lead to late delivery of final product.

5) Required very good management

   If management is not done properly then it would be very tough to get work done form the all team members.

### *2.4.10.5 Where to use Extreme Programming*

Extreme Programming works when Requirements are constant changing, high risk in there and rapid development of software is required then this model is used.

New challenges in projects, small groups of programmers are there. Moreover it is able to create automated tests. When direct involvement of customer is possible then this model is used.

Pair programming is the best approach for programming as they impart their knowledge among the team. So, better quality software is developed.

## 2.5 WHY ARE THERE SO MANY MODELS INTO EXISTENCE?

There are so many models available in market nowadays. All the models came into existence over a period of time since the birth of software development as different types of requirements come across during the development time. As the scenario of the customer's requirements changes over period of time, different kind of software development models come into existence.

Model can be chosen among different models depends on the project circumstances and requirements. A proper choice of a model can result in a more productive and efficient environment to develop a required software.

**The model is said to be the best model as if it aim for good, fast and cheap. It means every institute or organization who involve with software development, willing to develop the system with smoothness and at ease. For that model is required which is good in process, as fast to complete the system at the defined timeline and also cost effective so that cost would be control during the development process.** And so the developed system must be productive and efficient, meets the requirements of the customer.

## 2.6 COMPARISON OF AD-HOC, TRADITIONAL AND AGILE METHODOLOGY

| | Ad – hoc Methodology | Traditional Methodology | Agile Methodology |
|---|---|---|---|
| 1 | This methodology came into existence in 1960. | This methodology came into existence in 1970 and still in use. | This methodology started using in late 1990's. |
| 2 | There are no rules to follow with this methodology. | There are many rules, practices and documents with this methodology. | There are not much rules, and less documents. |
| 3 | As process goes on messy situation is created project might be failed. | As phases or practices are defined clearly so, process might lead to success. | As phases are defined but this is the new version of ad-hoc methodology, the project might lead to success. |
| 4 | This methodology is suitable for very small project. | This methodology is suitable for large project. | This methodology is suitable while updating or changes are required in the project. |
| 5 | Time taken is very less compared to other methodology. | Time taken for development is high. | Time taken is less as changes are being made with this methodology. |
| 6 | This methodology is cheap compared to other methodology. | This methodology is costly. | This methodology is not as costly as traditional methodology. |
| 7 | The project developed through this methodology is almost difficult to update or change. | The project developed through this methodology is not as difficult as ad-hoc as document is there. | The project developed through this methodology is not so difficult. |

**TABLE-1 COMPARISION BETWEEN AD-HOC, TRADITIONAL AND AGILE METHODOLOGY**

# 3. Comparison of Above Described SDLC Models

Waterfall Model is little hard to manage due to the rigidity of the model as each phase has specific deliverables and a review process. It works well for smaller projects where requirements are very well understood.

Prototype Model places more effort in creating the actual software instead of concentrating on documentation. This way, the actual software could be released in advance. Prototyping requires more user involvement and allows them to see and interact with a prototype allowing them to provide better and more complete feedback and specifications. The presence of the prototype being examined by the user prevents many misunderstandings that occur when each side believe the other understands what they said. The final product is more likely to satisfy the user's desire for look, feel and performance.

Spiral model is good for large and mission critical projects where high amount of risk analysis is required like launching of satellite.

Iterative and Incremental model is at the heart of a cyclic software development process. It starts with an initial planning and ends with deployment with the cyclic interactions in between. Easier to test and debug during a smaller iteration. Easier to manage risk because risky pieces are identified and handled during its iteration.

V-shaped Model has higher chance of success over the waterfall model due to the development of test plans during the life cycle. It works well for small projects where requirements are easily understood.

RAD Model is flexible and adaptable to changes as it incorporates short development cycles i.e. users see the RAD product quickly. It also involves user participation thereby increasing chances of early user community acceptance and realizes an overall reduction in project risk.

RUP Model is a complete methodology in itself with an emphasis on accurate documentation. It is proactively able to resolve the project risks associated with the client's evolving requirements. Less time is required for integration as

the process of integration goes on throughout the software development life cycle. The development time required is less due to reuse of components.

JAD Model can be successfully applied to a wide range of projects like new systems, enhancements to existing systems, System conversions, Purchase of a system etc.

SCRUM Model is successfully applied for simple and large scale projects. It is moderate in implementation and very high experts are required.

XP Model is applied for simple and small scale projects. To implements this SDLC high expert people are not required.

Comparison between different SDLC models in relation to their features like requirements, cost, resource control, risk involvement, changes incorporated, framework type, interface, reusability etc. is illustrated as below.

## 3.1 COMPARISON WATERFALL MODEL, PROTOTYPE MODEL, SPIRAL MODEL, ITERATIVE AND INCREMENTAL MODEL

| No | Model/features | Waterfall Model | Prototype Model | Spiral Model | Iterative and Incremental model |
|---|---|---|---|---|---|
| 1 | Specification of All the Requirements in the beginning | Yes | Not all and Frequently Changed | Not all and Frequently Changed | Not all and Frequently Changed |
| 2 | Project Cost | Almost as Estimated | Above Estimated Cost | Very Costly | Above Estimated Cost |
| 3 | Guarantee of Success | Low | Moderate | High | High |
| 4 | Required Expertise | Moderate | Moderate | High | Moderate |
| 5 | Overlapping Phases | No | Yes | No | Yes as Parallel development is there |
| 6 | Process | Heavyweight Process | Light weight Process | Heavyweight Process | Light weight Process |
| 7 | Framework type | Linear | Iterative | combination of Linear and iterative | Combination of Linear and Iterative |
| 8 | Rework cost | High | Not Low | High | Almost High |
| 9 | Testing | After coding phase Completed | After every iterative prototype model | At the End of Engineering Phase | After Every Iteration |
| 10 | Customer Involvement | Low | High, After Each Iteration | Low, After Each Iteration | High, After Each Iteration |
| 11 | Basic business Knowledge Required | Not much | Moderate | Not Much | Moderate |
| 12 | Suitable Project Size | Small Scale | Low to Medium Scale | Large Scale and Complex | Low to Medium Scale |
| 13 | Cost Control | Yes | No | Almost Yes | No |
| 14 | Simplicity | Simple | Moderate | Complex | Moderate |
| 15 | Risk Involvement | High | Low | Low | Low |

| | | | | | |
|---|---|---|---|---|---|
| 16 | Flexibility | Rigid | Much Flexible | Much Flexible | Much Flexible |
| 17 | Maintenance | Least Maintainable | Maintainable | Yes | Maintainable |
| 18 | Changes Incorporated | Difficult | Easily | Easily | Easily |
| 19 | Reusability | Least Possible | To some Extent | To some Extent | To some Extent |
| 20 | Documentation and Training | Necessary | Yes But Not Much | Yes | Yes But Not Much |
| 21 | Time Frame | Very Long | Long | Long | Long |
| 22 | Availability of Working Software | At the End of the Life Cycle | At the End of Every Iteration | At the End of Every Iteration | At the End of Every Iteration |
| 23 | Customized product | Least Possible | Possible | Possible | Much Possible |
| 24 | Customer Control over Administrator | Very Low | Yes | Yes | Yes |
| 25 | Required Team Creativity | No | Yes But Not Much | Yes But Not Much | Yes |
| 26 | Knowledge Transfer | No | Yes But Not Much | Yes But Not Much | Yes |
| 27 | Team size | Large Team | Small Team | Large Team | Not Large Team |
| 28 | Primary Objective | High Assurance | Rapid Development | High Assurance | Rapid Development |
| 29 | Implementation | Easy | Easy | Complex | Easy |
| 30 | Release Cycle | Big band(All Functionality at Once) | In Phases | Big band(All Functionality at Once) | In Phases |

**TABLE-2 COMPARISON OF SDLC MODELS_1**

## 3.2 COMPARISON OF V-SHAPED MODEL, RAD MODEL, RUP MODEL, JAD MODEL

| No | Model/features | V-shaped Model | RAD Model | RUP Model | JAD Model |
|---|---|---|---|---|---|
| 1 | Specification of All the Requirements in the beginning | Yes | Not all and Frequently Changed | Yes | Not all and Frequently Changed |
| 2 | Project Cost | Almost as Estimated | Almost as Estimated | Almost as Estimated | Expensive |
| 3 | Guarantee of Success | Moderate | Very Good | Very High | High |
| 4 | Required Expertise | Moderate | Moderate | High | Very High |
| 5 | Overlapping Phases | No | Yes | No | Yes |
| 6 | Process | Heavyweight Process | Light weight Process | Heavyweight Process | Lightweight Process |
| 7 | Framework type | Non linear | Prototype and Iterative | Iterative and Incremental | Incremental |
| 8 | Rework cost | High | Not very High | High | High |
| 9 | Testing | After completion of Each Iteration | After Completion of Coding | At the Construction Phase | After Coding Phase |
| 10 | Customer Involvement | Low | High | High, After Each Iteration | Continuous |
| 11 | Basic business Knowledge Required | Not Much | Required | Required | Very Much |
| 12 | Suitable Project Size | Large Scale | Low to Medium Scale | Small as well as Large Scale | Small as well as Large Scale |
| 13 | Cost Control | Yes | No due to Urgent Requirement | No | Almost No |
| 14 | Simplicity | Moderate | Simple | Complex | Simple |
| 15 | Risk Involvement | Not High | Very Low | Low | Low |
| 16 | Flexibility | Little Flexible | Very Flexible | Very Flexible | Very Flexible |
| 17 | Maintenance | Little Maintainable | Easily Maintainable | Hard to Maintain | Maintainable |

| | | | | | |
|---|---|---|---|---|---|
| 18 | Changes Incorporated | Difficult | Easily | Easily | Easily |
| 19 | Reusability | Little Possibility | Yes | Yes | Yes |
| 20 | Documentation and Training | Yes | Limited | Very Limited | Yes |
| 21 | Time Frame | Long | Short | Long | Moderate |
| 22 | Availability of Working Software | At the End of the Life Cycle | At the End of the Life Cycle | At the End of the Life Cycle | At the End of Every Iteration |
| 23 | Customized product | Least Possible | Possible | Possible | Possible |
| 24 | Customer Control over Administrator | Low | Yes | Yes | Very Much |
| 25 | Required Team Creativity | No | Yes | Yes | Yes |
| 26 | Knowledge Transfer | No | Yes | Yes | Yes |
| 27 | Team size | Small Team | Small Team | Large Team | Large Team |
| 28 | Primary Objective | High Assurance | Rapid Development | High Assurance | Rapid Development |
| 29 | Implementation | Easy | Easy | Complex | Moderate |
| 30 | Release Cycle | Big band(All Functionality at Once) | In Phases | Big band(All Functionality at Once) | In Phases |

**TABLE-3 COMPARISON OF SDLC MODELS_2**

## 3.3 COMPARISON OF SCRUM MODEL, EXTREME PROGRAMMING MODEL

| No | Model/features | Scrum Model | Extreme Programming Model |
|---|---|---|---|
| 1 | Specification of All the Requirements in the beginning | Not all and Frequently Changed | Not all and Frequently Changed |
| 2 | Project Cost | Almost as Estimated | Almost as Estimated |

| | | | |
|---|---|---|---|
| 3 | Guarantee of Success | High | High |
| 4 | Required Expertise | Very High | Moderate |
| 5 | Overlapping Phases | Yes | Yes |
| 6 | Process | Light weight Process | Light weight Process |
| 7 | Framework type | iterative and incremental | Iterative |
| 8 | Rework cost | High | High |
| 9 | Testing | After Coding Phase | automated testing while coding |
| 10 | Customer Involvement | High | Continuous |
| 11 | Basic business Knowledge Required | Very Much | Very Much |
| 12 | Suitable Project Size | Large Scale | almost small scale |
| 13 | Cost Control | No | No |
| 14 | Simplicity | Simple | Simple |
| 15 | Risk Involvement | Not High | Not High |
| 16 | Flexibility | Flexible | Very Flexible |
| 17 | Maintenance | Maintainable | Easily Maintainable |
| 18 | Changes Incorporated | Easily | Easily |
| 19 | Reusability | Yes | Yes |
| 20 | Documentation and Training | Limited | Limited |
| 21 | Time Frame | Moderate | Short |
| 22 | Availability of Working Software | At the End of Every Iteration | At the End of Every Iteration |
| 23 | Customized product | Possible | Possible |
| 24 | Customer Control over Administrator | Yes | Very Much |
| 25 | Required Team Creativity | Yes | Yes |
| 26 | Knowledge Transfer | Yes | Yes |
| 27 | Team size | Large Team | Small Team |

| | | | |
|---|---|---|---|
| 28 | Primary Objective | Rapid Development | Rapid Development |
| 29 | Implementation | Moderate | Easy |
| 30 | Release Cycle | In Phases | In Phases |

**TABLE-4 COMPARISON OF SDLC MODELS_3**

# 4. Incorporate Lessons Learned with SDLC

In today's world, there are lots of Software Development Life Cycle Models. We have seen some of them earlier. We have seen the analytical study of these models also in the previous section. All the SDLC models have their own advantages and limitations. According to the requirements of the user the SDLC models is chose to develop the software.

Some of them need the entire requirement earlier and some of them require some of the requirements in starting stage. Some models respond to changes easily and some of them are very rigid in nature to corresponding changes. There are lots of aspect basis on researcher made analytical survey.

There are mainly three types of SDLC models. Out of which two types of model are used by software companies to develop software.

Researcher has find out that when there is need of completely new software based on new technology at that time "Traditional models" are used to develop software. When there is need to accommodate changes in the current software system at that time "Agile models" are required. Mostly this kind of scenario is there in the software development company.

In traditional models, there are lots of models available. From which researcher has taken Waterfall model, V-shaped model, Spiral model, Rational unified Process model, etc.

## 4.1 SPIRAL MODEL

Among them Spiral model is best suitable to use when user is not specified about their needs very earlier in the project development life cycle. Requirements keep on changing during the development process of the software. When completely new technology is used for the new software at this time spiral model is used.

It provides risk management as a special phase. So risk analysis and risk management is done in this phase. As large scale system contains large risk throughout the development life cycle, so it has to manage.

The spiral model uses prototyping as a risk reduction mechanism but more important, enables the developer to apply the prototyping approach at any stage in the evolution of the product. It manages the systematic stepwise approach suggested by the classic life cycle but incorporates it into an iterative framework that more realistically reflects the real world.

So, it can be concluded that spiral model is used for “large, expensive and more over for complex software”.

The main limitation is that if a major risk in not uncovered and managed, problems will undoubtedly occur. It will take quite long time to develop the software under this approach.

## 4.2 OBJECT MODELING TECHNIQUE[35]

### Introduction:

There are many ways to look at a problem to be solved using software-based solution. One widely used approach to problem solving is Object Oriented viewpoint.

We live in a world of objects. These objects exist in nature, in human made entities, in business and in the products that we use. Objects can be categorized, described, organized, combined, manipulated, and created. Therefore, object oriented view would be proposed for the creation of computer software – an abstraction that enables us to model the world in ways that help us to better understand and navigate it.

An object-oriented approach to the development to software was first proposed in the late 1960s.

The problem domain is characterized as a set of objects that have specific attributes and behaviors. The objects are manipulated with a collection of functions, which can be called methods, operations and services. Object can communicate with one another through a message protocol.

[35] Object-Oriented Modeling and Design by James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen.- PHI publication

An object encapsulation both data and the processing that is applied to the data. This important characteristic enables classes of objects to be built and inherently lead to libraries of reusable classes and objects.

The already defined classes and objects can be used in the further development of the software. This concept is known as Reusability. This concept is most important to develop software which is based on iterative and incremental approach.

This reusability is a critically important attribute of modern software engineering, the object modeling technique is attractive to many software development organizations.

Object oriented technology lead to reuse, and reuse of program components lead to faster software development and higher quality programs. Object oriented software is easier to maintain because its structure inherently decouple. This leads to fewer side effects when changes have to be made and less frustration for the software engineer and the customer. In addition, object-oriented system are easier to adapt and easier to scale for i.e. large system can be created by assembling reusable subsystem.

## 4.3 OBJECT ORIENTED CONCEPTS[36]

Any discussion of object-oriented software engineering must begin by addressing the term object-oriented. It means organize software as a collection of discrete objects that incorporated both data structure and behavior. This is in contrast to conventional programming in which data structure and behavior are loosely connected. There are generally four aspect of Object Oriented: Identity, classification, polymorphism and inheritance.

Identity means that data is quantized into discrete, distinguishable entities called objects. A paragraph in a document, a window on my workstation, and the white queen in a chess game are examples of objects. Object can be concrete such as a file in a file system or conceptual, such as a scheduling

[36] Object-Oriented Modeling and Design by James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen.- PHI publication

policy in a multiprocessing operating system. Two objects are distinct even if all their attribute values are identical for i.e. two different tables have same color and having same size then it can't be called that they are one. Thought they have same values for their attributes, they are different object.

Classification means that objects with the same data structure (attributes) and behavior (operation) are grouped into a class. Paragraph, window and Chess Piece are example of classes. A class is an abstraction that describes properties important to an application and ignores the rest. Any choice of classes is arbitrary and depends on the application.

Each class describes a possibly infinite set of individual objects. Each object is said to be an instance of its class. Each instance of the class has its own value for each attribute but shares the attribute names and operations with other instances of the class. An object contains an implicit reference to its own class; it knows what kind of thing it is. Example is given below.

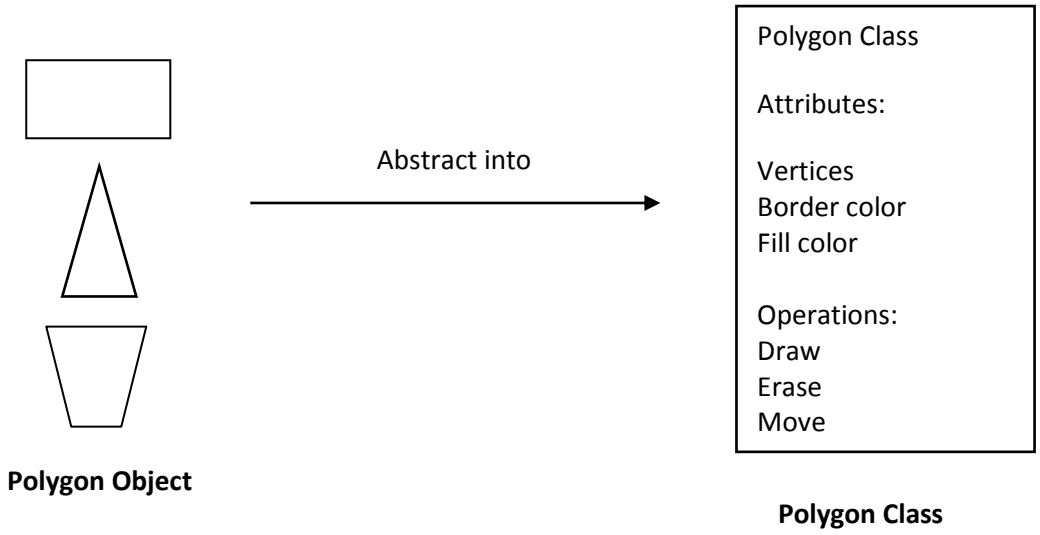



**FIGURE-17 OBJECT AND ITS ATTRIBUTES**

Polymorphism means that the same operation may behave differently on different classes. The move operation may behave differently on different classes. It moves a triangle object towards right and rectangle object towards left. An operation is an action or transformation that an object performs or is

subject to. Right-justify, display, delete are examples of operations. A specific implementation of an operation by a certain class is called a method. Because an object-oriented operator is polymorphic, it may have more than one method implementing it.

Inheritance is the sharing of attributes and operation among classes based on a hierarchical relationship. A class can be defined broadly and then refined into successively finer subclasses. Each subclass incorporates or inherits all of the properties of its super class and adds its own unique properties. The properties of the super class need not be repeated in each subclass. The ability to factor our common properties of several classes into a common super class and to inherit the properties from the super class can greatly reduce repetition within designs and programs and is one of the main advantages of an object-oriented system. For example, Scrolling Window and FixedWindow are subclasses of Window. Both subclasses inherit the properties of Window, such as visible region on the screen. Scrolling Window adds a scroll bar and an offset.

## 4.4 OBJECT-ORIENTED THEMES[37]

There are several themes underlying object-oriented technology. Although these themes are not unique to object-oriented systems, they are particularly well supported in object-oriented systems.

### Abstraction

Abstraction consists of focusing on the essential, inherent aspects of an entity and ignoring its accidental properties. In system development, this means focusing on what an object is and does, before deciding how it should be implemented. Use of abstraction preserves the freedom to make decisions as long as possible by avoiding premature commitments to details. Most modern languages provide data abstraction, but the ability to use inheritance and

[37] Object-Oriented Modeling and Design by James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen.- PHI publication

polymorphism provides additional power. Use of abstraction during analysis means dealing only with application-domain concepts, not making design and implementation decisions before the problem is understood. Proper use of abstraction allows the same model to be used for analysis, high-level design, program structure, database structure, and documentation. A language-independent style of the design defers programming details until the final, relatively mechanical stage of development.

**Encapsulation:**

Encapsulation, which also known information hiding consists of separating the external aspects of an object, which are accessible to other objects, from the internal implementation details of the object, which are hidden from other objects. Encapsulation prevents a program from becoming so interdependent that a small change has very big ripple effects. The implementation of an object can be changed without affecting the application that uses it. One may want to change the implementation of an object to improve performance, fix a bug, and consolidate code, of for porting. Encapsulation is not unique to object-oriented languages but the ability to combine data structure and behavior in a single entity makes encapsulation cleaner and more powerful than in conventional languages that separate data structure and behavior.

**Combining Data and Behavior:**

The caller of an operation need not consider how many implementation of a given operation exist. Operator polymorphism shifts the burden of deciding what implementation to use from the calling code to the class hierarchy. For example, non-object oriented code to display the contents of a window must distinguish the type of each figure, such as a polygon, circle or text, and call the appropriate procedure to display it. An object-oriented program would simply invoke the draw operation each figure; the decision of which procedure to use is made implicitly by each object, based on its class. It is unnecessary to repeat the choice of procedure every time the operation is called in the application program. Maintenance is easier, because the calling code need not be modified when a new class is added. In an object-oriented system, the data structure hierarchy is identical to the operation inheritance hierarchy.

**Sharing:**

Object-oriented techniques promote sharing at several different levels. Inheritance of both data structure and behavior allows common structure to be shared among several similar subclasses without redundancy. The sharing of code using inheritance is one of the main advantages of object-oriented languages. More important than the saving in code is the conceptual clarity from recognizing that different operation are all rally the same thing. This reduces the number of distinct cases that must be understood and analyzed.

Object-oriented development not only allows information to be shared within an application, but also offers that prospect of reusing design and code on future projects. Although this possibility has been overemphasized as a justification for object-oriented technology, object-oriented development provides the tools, such as abstraction, encapsulation, and inheritance, to build libraries of reusable components. Object-oriented is not a magic formula to ensure reusability, however. Reuse does not just happen; it must be planned by thinking beyond the immediate application and investing extra effort in a more general design.

## 4.5 EMPHASIS ON OBJECT STRUCTURE, NOT PROCEDURE STRUCTURE[38]

Object-oriented technology stresses specifying what an object is, rather than how it is used. The uses of an object depend highly on the details of the application and frequently change during development. As requirements evolve, the feature supplied by an object are much more stable than the ways it is used, hence software system built on object structure are more stable in the long run. Object-oriented development places a greater emphasis on data structure and a lesser emphasis on procedure structure than traditional functional-decomposition methodologies. In this respect, object-oriented development is similar to information modeling technique used in database

[38] Object-Oriented Modeling and Design by James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen.- PHI publication

design, although object-oriented development adds the concept of class-dependent behavior.

## 4.6 OBJECT-ORIENTED METHODOLOGY[39]

We present a methodology for object-oriented development and a graphical notation for representing object-oriented concepts. The methodology consists of building a model of an application domain and then adding implementation details to it during the design of a system. We call this approach the Object Modeling Technique (OMT). The methodology has the following stages:

### 1) Analysis

Starting from a statement of the problem, the analyst builds a model of the real-world situation showing its important properties. The analyst must work with the requestor to understand the problem because problem statements are rarely complete or correct. The analysis model is a concise, precise abstraction of what the desired system must do, not how it will be done. The object in the model should be application domain concepts and not computer implementation concepts such as data structure. A good model can be understood and criticized by application experts who are not programmers. The analysis model should not contain any implementation decisions. For example a Window class in a workstation windowing system would be described in terms of the attributes and operations visible to a user.

### 2) System Design

The system designer makes high level decisions about the overall architecture. During system design, the target system is organized into subsystem based on both the analysis structure and the proposed architecture. The system designer must decide what performance

[39] Object-Oriented Modeling and Design by James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen.- PHI publication

characteristic to optimize, choose a strategy of attacking the problem, and make tentative resource allocations. For example, the system designer might decide that changes to the workstation screen must be fast and smooth even when windows are moved or erased, and choose an appropriate communications protocol and memory buffering strategy.

**3) Object Design**

The object designer builds a design model based on the analysis model but containing implementation details. The designer adds details to the design model in accordance with the strategy established during system design. The focus of object design is the data structures and algorithms needed to implement each class. The object classes from analysis are still meaningful, but they are augmented with computer domain data structure and algorithms chosen to optimize important performance measures. Both the application domain objects and the computer-domain objects are described using the same object-oriented concepts and notation, although they exist on different conceptual planes. For example, the Window class operations are now specified in terms of the underlying hardware and operating system.

**4) Implementation**

The object classes and relationships developed during object design are finally translated into a particular programming language, database, or hardware implementations. Programming should be a relatively minor and mechanical part of the development cycle, because all of the hard decisions should be made during design. The target language influences design decisions to some extent, but the design should not depend on fine details of a programming language. During implementation, it is important to follow good software engineering practice so that traceability to the design is straight forward and so that the implemented system remains flexible and extensible. For example, the Window class would be coded in a programming language, using calls to the underlying graphics system on the workstation.

Object-oriented concepts can be applied throughout the system development life cycle, from analysis through design to implementation. The same classes

can be carried from stage to stage without a change of notation, although they gain additional implementation details in the later stages. Although the analysis view and the implementation view of Window are both correct, they serve different purposes and represent a different level of abstraction. The same object-oriented concepts of identity, classification, polymorphism, and inheritance apply through the entire development cycle.

Though some classes are not part of analysis but they are introduced as part of the design or implementation. For example, data structures such as trees, hash tables, and linked lists are rarely present in the real world. They are introduced to support particular algorithm during design. Such data structure objects are used to implement real world objects within a computer and do not derive their properties directly from the real world.

## 4.7 CONCLUSION TO CHOOSE THE NEW MODEL

### 4.7.1 Advantages of Object-Oriented Methodology

**1) Reduced Maintenance:**

The main goal of object-oriented development is the assurance that the system will have a very long life with very smaller amount of maintenance. Most of the processes within the system are encapsulated; the methods may be reused and incorporated into new methods.

**2) Real-World Modeling:**

Object-oriented system uses to model the real world in a more complete fashion than traditional methods do. Objects are organized into classes of objects, and objects are associated with methods. The model is based on objects, rather than on data and processing.

**3) Improved Reliability and Flexibility:**

Object-oriented system is more reliable than traditional systems, because new behaviors can be built from existing ones. As objects can be dynamically called and accessed, new objects may be created at any time. The new objects may inherit data attributes from one, or many other objects. Behaviors

may be inherited from super classes and new methods may be added without effecting exiting systems functions.

**4) High code Reusability:**

When new object is created, it will automatically inherit the data attributes and characteristics of the class from which it was created. The new object will also inherit the data and methods from all superclass. New behavior is also added to the newly created object.

### 4.7.2 Advantages of Spiral Model

1) Spiral life cycle model is one of the most flexible SDLC models. Development phases can be determined by the project manager, according to the complexity of the project.

2) Project monitoring is very easy and effective as it is done in each phase of each iteration. This monitoring is done by experts or by concerned people. This makes the model more transparent.

3) Risk management is the key feature of this model, which makes it more attractive compared to other models.

4) If changes are introduced at later stage in life cycle, coping with these changes isn't a very big problem for the project manager.

5) It is suitable for high risk projects, where business needs may be unstable.

6) A highly customized product can be developed using this model.

7) Since the prototype building is done in small fragments or bits, cost estimation becomes easy and the customer can gain control on administration of the new system.

8) As the model continues towards final phase, the customer's expertise on new system grows, enabling smooth development of the product meeting client's needs.

Lots of advantages of the Object-oriented Methodology are there. Reusability is highly advantageous in current scenario of system development. Although

all the advantages of the Object-oriented Methodology can be the key characteristics of the system developed in the today's world scenario.

The system developed using Object-oriented, has least maintenance cost with lots of reliability and flexibility. Moreover, Object-oriented Methodology is reliable to develop software based on real world problem.

As there is in market, lots of new versions of the system is created and launched in market at different interval of time. If this incremental development is done using Object-oriented methodology then it would be very easy to develop the new versions of the system.

As spiral model is used with the large, complex and expensive software, the incremental development is there. Requirements are very uncertain and constant changes is there while the development of the software.

Risk assessment and management is also a good feature of spiral model, which can lead to a successful and secure development of the system.

The feature of both the approaches, spiral model and object-oriented model can be used in a single approach to get all the advantages of these approaches. This newly develop model has all the feature and quality of both approach.

Risk assessment and management is the best feature of the spiral model. Constant changes are also easily incorporated with the spiral model. But it would be very easy if reusability is added with this approach.

As cost is very high of spiral model but if it combines with Object-oriented approach, which will lead to minimize the maintenance cost of the system.

As most of the conventional approach are least flexible, though spiral model is reliable and least flexible as compared to object-oriented approach. If spiral approach and Object-oriented approach combine to gather the new approach will have the characteristic of flexibility and reliability.

So, researcher wants to combine both the approach in a single approach which can lead to develop large and complex system with reusability concept. This can facilitate the developer a fast development in comparison with the other conventional approach.

There is component based development which is based on spiral model and Object oriented model. Below is the description of component based development model.

## [40]4.8 COMPONENT BASED DEVELOPMENT

Object-oriented technologies provide the technical frame work for a component based process model for software engineering. The object oriented paradigm emphasized the creation of classes that encapsulated both data and the algorithm used to manipulate the data. If properly designed and implemented, object-oriented classes are reusable across different application and compute-based system architectures.

The component based development model incorporates many of the characteristic of the spiral model. It is evolutionary in nature, demanding an iterative approach to the creation of software. However, the component-based development model composes applications from pre-packaged software components.

The engineering activity begins with the identification of candidate classes. This is accomplished by examining the data to be manipulated by the application and the algorithm (method) that will be applied to accomplish the manipulation. Corresponding data and algorithms are packaged into a class.

Classes created in past software engineering projects are stored in a class library or repository. Once candidate classes are identified, the class library is searched to determine if these classes already exist. It they do, they are extracted from the library and reused. If a candidate class does not reside in the library, it is -engineered using object-oriented methods. The first iteration of the application to be built is then composed, using classes extracted from the library and any new classes built to meet the unique needs of the application. Process flow then returns to the spiral and will ultimately re-enter the component assembly iteration during subsequent passes through the engineering activity.

[40] Software Engineering- A practitioner's Approach, Roger S. Pressman. McGrawHill Publication.

The component-based development model leads to software reuse, and reusability provides software engineers with a number of measurable benefits.

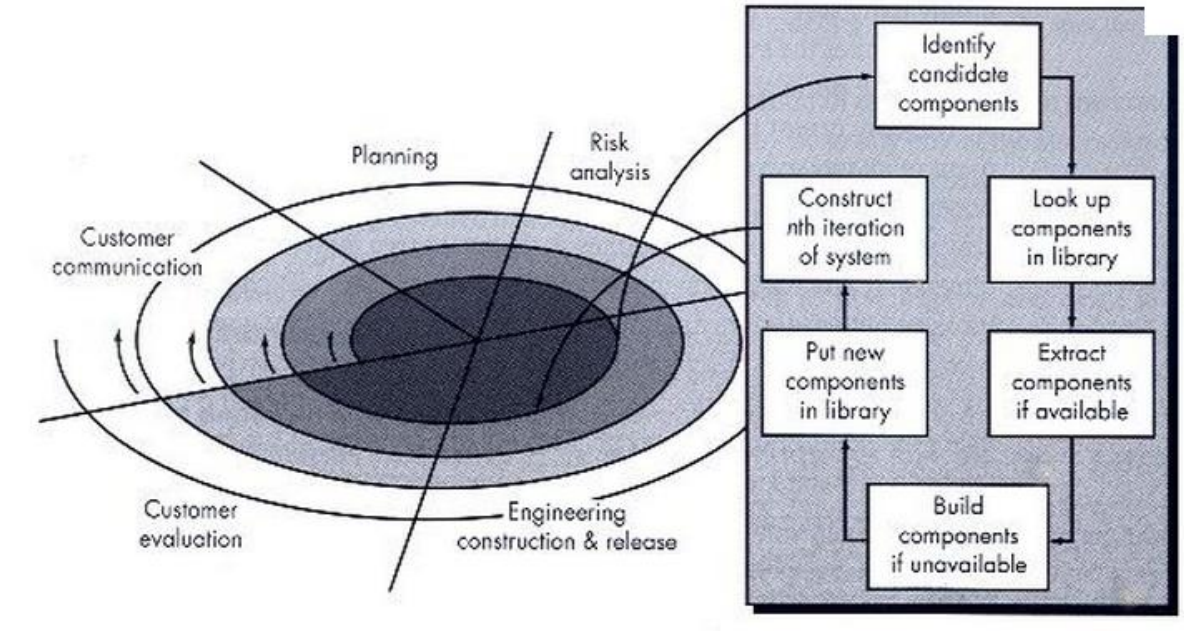



**FIGURE-18 COMPONENT BASED MODEL**

### 4.8.1 Advantages of Component Based Model

1) Since it component based model so, the changes could be made easily in any component.
2) It is iterative in nature, after each iteration user can give their feedback.
3) Risk analysis can be done with this model.
4) Software based on this model can be reused by making considerable changes.
5) Object oriented approach is used in this model so it supports reusability concept.
6) Highly reliable software can be created.
7) User feedback is quite useful after iteration which could lead to the successful system which is desired by the user or owner.

According to researcher point of view, this model is quite useful. So, researcher has tried to modify this model which could be more useful for software development.

## 4.9 INTRODUCTION TO LESSONS LEARNED

To understand what Lessons Learned is we must have to understand what is Lessons? Lessons can be derived from any activity or task which is being done during any process. For i.e. if a person want to reach to the destination called B from the origin place A. There are two ways to reach to the destination called p1 and p2. He started his journey with the p1. As he moves on he found lots of hurdles during the journey. If he takes p2 path then his journey would be smooth. But he was unaware about the p2. So, now he could share this information to the people around him who are using that road, so they would not face the difficulty. So, this is the lesson to reach from A to B destination, path p2 is the best choice which led the smooth and hurdle free journey. This is the simple example of how lessons can be found.

So, in technical terms lessons are a product of operations, exercises, training, experiments, and day-to-day staff work. During the course of our activities most of us will recognize ways of doing things more easily or efficiently that can be passed on to our colleagues and successors to help them avoid problems and do even better than we did before.

The word Lessons Learned is generally used to explain the act of learning from experience to achieve improvements. It is an activity used by people to get benefit in current on-going activity from past activity experience. Past experience would be “Positive or Negative”. Negative experience is in such sense that any problem occurs during development process and solution must have to be found to eliminate problem. Positive experience is the experience that works well or best for project. As we all that experience is the best teacher, we can learn from the past project experience.

The idea of lessons learned in an institute is that by using a formal approach to learning, individual and the whole institute or organization can minimize the

risk of repeating mistakes and it would increase the chance of success. This means lessons learned approach will reduce failure risk, and improve operational effectiveness and also increase cost efficiency.

Lessons learned is not just learning from experience but used to justify changes that will lead to improved performance. So, Lessons learn can be defined as follows:

“The purpose of a Lessons Learned approach is to learn efficiently from experience and to provide validated explanation for revising the existing way of doing things, to improve performance, both during the course of an operation and for next operation.”

In project development organization, Lessons learned approach must be considered as a complete activity and it must be executed properly. It is very important process which is mainly consist of taking information from past projects and putting it to use in upcoming projects or next iteration of the same project. This is done by extracting the relevant information and storing it for future use and analysis.

The main goal of the lessons learned process is to provide information that will enhance the efficiency of future projects in term of cost, labor, and time. The information collected during the project development should be held, in digital format or in paper format. It should be handy to the project’s team members. These people, as individuals, cannot transform the information they gathered during course of project development into considerable format, so that anyone can use it. In many cases, in software development organization, the team members leave the organization upon the completion of the project which leads to the possible loss of significant information when the project is complete.

Of course it is not like that the team members would leave the organization only after completion of the project. They can also leave the organization in between of the project development process also. Some of the organizations document the positive and negative issues at the end of the project or we can say prepare lessons learned document at the end of the project. If lessons learned documents are prepared at the end of the project development than

there must be loss of information of positive and negative issues raised during project development due to leaving of organization by any team member of the project. So, lessons learned process should be continuous process throughout the project development. This is the first reason why lessons learned approach is emerged with SDLC.

Here is another reason to include it with SDLC, throughout the project life cycle if any issue rose during the project development at any phase; can team member remember it till the end of the project development? This is the big question. Organization will be in loss of information of such issues if team member forget as the issues are not fresh. They become old. Team members even don't remembers which kind of remedial action had been taken to solve the issue. So, for not losing the information, lessons learned process should be continuous process throughout the project development cycle.

"Lessons learned document should be one of the deliverable of the each phase of the project development life cycle." To make this happen, organization should have a lessons learned team, working as project team in co-operation with project development team.

This requires lessons should be meaningful and proper format so that appropriate authority can use it. It must give clear understating of the issues related to the system development phased during project development phases.

Lessons learned should be used as a phase as it would be proved as very important phase of the software development life cycle. Here, researcher has tried to incorporate lessons learned approach with the software development life cycle in both ways as a continuous process and as a phase at the end of the SDLC.

Successfully implement this lessons learned approach, active participation of lessons learned team, project management team, project development and technical team and administrative people should be there throughout the project development. Personnel must be encouraged to share new lessons from their prior or ongoing project and also apply already given or published lessons to their areas of responsibility.

There are three basic "**STEPS OF LESSONS LEARNING"** with respect to Software Development Life Cycle Process.

### 4.9.1 Identification

In this step, collect learning from each phase of the SDLC Process. During each phase identify each experience which can be helpful for next iteration of this running process or it can be the different project. The experience can be good or bad. The good experience can lead the project team that they are doing right or on a proper track of development life cycle. And from the bad experience the team can be aware of taking the wrong path of the software development.

### 4.9.2 Action

To take action to change existing way of doing things based on the learning. From good experience, the project team member can go along with the path for success. But from bad experience the project team member can learn to change their path to get success. It means whatever action taken in previous iteration or previous project, which was not satisfactory for that stage could be changed. So, from that bad experience, team member can learn to change the existing way of doing the things which could lead to the success.

Previous project or previous iteration's documents are reviewed by the project team. They can learn to implement action differently rather than the previous one if any action had lead the project development procedure inefficient to some extent.

### 4.9.3 Harmonization (To make available)

Communicate the change so that related parts of the organization can benefit from the learning. It is the most important step of the lessons learned approach. It is most important to aware the people about the changes applied

to the conventional or previously done task. People of the organization must aware about changes so they can also get benefited from the action taken.

To do so, to introduce this new action to another people of the institute training can be arranged. Or new staff can get informed by the old or senior staff through the means of mails, newsletter, etc.

Project team members/stakeholders within an organization need to be involved in learning lessons process for the lessons learned approach to be successful. It seems that many personnel under the project team may have impression that they don't require to learn anything new as they already know everything or participate in preparing lessons learned document. This is the general scenario of every organization. But this is not the worth for the organization as the computer science is very young profession, so changes are continuously there with this industry.

Lessons are not learned until something changes are made at the time of implementation. Stakeholders are personnel who need to change to way of implementation of any action as they are closely related to the development of the project. Stakeholders are the learner. Stakeholders are the only personnel who will be aware of potential lessons like observations and lessons identified since they are the most closely involved with the issue.

Each team members or stakeholders should have capability to identify the issue and must submit it in a proper manner. Otherwise it can't be known by other personnel. So the potential issues must be shared by team members.

No team members should have authority to implement any changes with respect to any issue without the permission of any of the respected higher authority. As the issues which changes the way to do any activity must be submitted to the higher authority if they permit then team member should go ahead otherwise team member should not implement any changes.

## 4.10 WHY LESSON'S LEARNED IMPORTANT

Each team members under the project thinks that they know better. They think like that they have done a better schedule management or better budgeting, more communication, and spent time on collecting requirements,

etc. All these things are about how to do the work, not what to work on. Planning is just like talking about how things get done or working on how things get done or does not get done. But planning may be failed. Generally, this could be the scenario that project development does not cope with the planning, as actual situation is different than planning. This is one of the reasons why team members hate planning and planning is doing nothing and we all like doing something.

But doing action without proper planning is root cause of the problem. The idea behind this is that we all think that action is better than thought or discussion or planning. But before doing anything there must have to plan out about the task; there are multiple ways to complete the same task. From which, the best way should be chosen. To find out which way to choose or which approach is best to complete the task lessons learned is used.

Everything learned from previous projects, whether they were successes or failures, those learning can teach important lessons. All the team members can learn important lessons. If lessons were genuinely learned from past projects then the same mistakes would not be repeated on different projects or next iteration of the same project. Projects within an organisation would then be more consistently delivered on time, within decided budget and to the customer's complete satisfaction.

When project environment are challenging with multi-functional team and they can be culturally and geographically diverse from each other. Budgets are usually tightly constrained and business requirements frequently change during development time. Project owner may not have effective communication to the development team and there can be different department which are not well integrated-with the result the similar mistakes are often repeated.

In such scenario, it would advisable not to repeat mistake again and again to save money and time of project owner and stakeholder of the project. The technological infrastructure should be available to support the transfer of knowledge across the whole team and all the departments involved in a project. So, in such environment lessons could be learnt.

We consider a case study through which we can understand why lessons learned approach is necessary.

It is a practise of software Development Company that any project is under construction has a weekly meeting about that project. The meeting is based on the status of the project going on. Management quickly review the status of the project, go over any big issues and review and approve changes if it is needed.

As the project goes on in development stage, these meeting also continue in weekly manner. It is a practise to held weekly meeting in every project. But at later stage of software development it is not required. It should go to once in a month. The main reasons seem to be that there are too many people in the meeting, the topics are repetitive, the board gets weekly status reports, and that the meeting is boring and waste of time.

Now suppose these weekly meetings will take 1/2 hour when there is no issue. While monthly meeting with the management will last for 45 minutes or may be of one hour. So, by calculating average time, personnel can get extra time to give their performance. Employee would save almost about 50% of their time.

As the development goes on meeting time should decreased from 45 minutes to 30 or less than 30 minutes. This time can be utilized by the team members in a constructive manner which would be beneficial to the organization.

So, this is the scenario where lessons learned approach is useful. Project manager should learn from the lessons learned and understand how to utilize the personnel time for the development of the project. By changing in the conventional approach, he can do something beneficial to the organization.

These lessons learned are useful. The time spent in doing the work better is time well spent. Get the lessons learned right at that time the first time is cheaper and easier than doing it now and fixing it later. It means whatever the changes in the activity are; it should be implemented for betterment of the project.

So, lessons learned from the past projects are really very useful and can prevent problems later down the line; organisations should create a lessons

learned culture where people not only take the trouble to learn from past projects, but actually to learn something useful from past project.

Many project teams conduct a “lessons learned” review at the end of the project. It is useful but only for short duration or small project. As the project completes they can put their good or bad experience in document form at the end of the project. Here with small project small no of personnel are involved with the project. So, it would be easy for them to document their experience at the end of the project life cycle.

What about large project or the project which last for one to two years? With such projects the no of personnel involved are also large. There are large no of team members to accomplish different task of the same project. Some of the personnel left the project in the mid of the project and new personnel would join the project. In such scenario it would be impossible to document their experience. Another thing about large project is that after experiencing any good or bad experience in the starting or at the mid of the project development team member would forget about it at the time of lessons learned in the end of project development.

So, in such scenario, it would be beneficial for the organization, to document all the good and bad experience at the end of each phase of the development. The value lays in make concrete the path for future projects, so they will experience less problems and setbacks. There is no need to wait until the entire project finishes. Issues could be captured incrementally throughout the project, or very soon after it's over, while the issues are still fresh.

## 4.11 BENEFITS OF LESSON’S LEARNED

1) Management will get the complete scenario of what went well and what didn’t go well during the project development life cycle.
2) Stakeholders or team members can learn lessons from past experiences recorded in the lessons learned of the same type of project or from previous iteration of the same project.

3) For any problems that went largely unresolved, for instance, consider preventative measures that could help people work through or avoid the problems in the future.

4) It might involve tools to speed up the work, checklists to make sure people don't skip important steps, and solutions for difficult problems. Like if you found components that were supposed to work together but didn't, and someone found a solution or a workaround, record that information to help people prevent to solve similar problem in the future.

5) You can use your repository[41] of lessons learned to help your customer service or technical support personnel solve problems in a just-in-time fashion, for example. If your staff or colleagues need to address complex issues on the fly, or troubleshoot technical issues very quickly over the telephone, they would need fast access to solutions for similar situations that were addressed in the past.

6) By capturing issues incrementally during project development, you will achieve far greater long-term success than by ignoring or forgetting problems, or by simply moving on when a project or phase ends.

7) Organization will get benefit in monetary aspects as some issues are already solved before arises.

8) It will also help team members in saving their time by guiding them what to do and what to not do. So, resource will be utilized in proper and right manner.

So, above are the benefits to have lessons learned approach throughout the whole life cycle of the project.

## 4.12 LESSONS LEARNED ABILITY

The lessons learned ability provides organization structure, process, and tools which are necessary to capture, analyse and take remedial action on any issue and to communicate and share results to achieve improvement.

[41] Here repository means Lessons learned database

Lessons learned is an approach to share information about what had happened good or bad events with respect to past project. There are main three elements which are needed to support lessons learned approach to get success are structure, process, tools to support process.

## 4.13 STRUCTURE OF LESSON'S LEARNED TEAM

Lessons learned team should be there in all the organization for developing project. All though lessons learned approach should be adapted by all the organization who is involve in manufacturing, imparting services, or any business. To deal with the lessons learned should be managed by this team. It means lessons learned team should be the ultimately responsible for lessons learned approach. Identify issue/observation; get information about issue or observation, how to deal with them, how to store and where to store them, how to make lessons learned easily available to required personnel all must be decided by lessons learned team.

So, there are lots of responsibilities of lessons learned approach. To manage these responsibilities there must be personnel appointed to do particular task. The hierarchy is defined below:

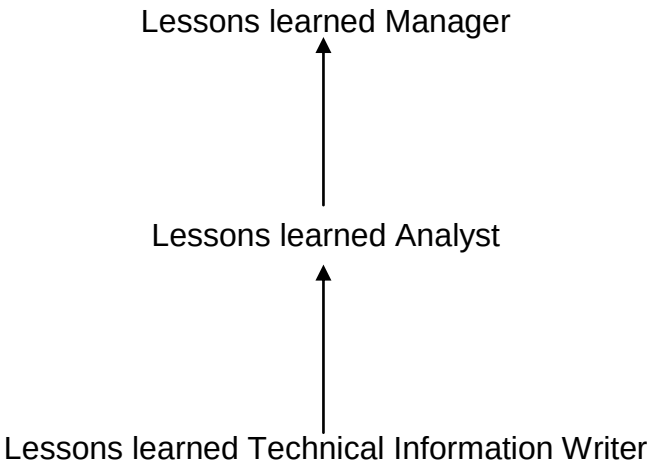



**FIGURE-19 HIERARCHY FOR LESSONS LEARNED**

Organization must have Lessons learned manager, lessons learned analyst, lessons learned technical information writer. Depending upon the company and depending upon the size of the project no of personnel can be varying.

### 4.13.1 Roles and Responsibilities of Lessons Learned Team

#### *4.13.1.1 Lessons learned Manager (1)*

Lessons learned manager plays important role in Lessons learned team. Generally, lessons learned manager's position is single. He/she is the leader of the lessons learned team.

Lessons learned manager works with the whole lessons learned team as well as with the project manager. If any issues arise during project development, project manager should get solution from lessons learned manager if he/she has. Lessons learned manager should be the ultimate responsible for making the information available to the project management team. Lessons learned manager should remain present during any meeting arranged during project development process.

Lessons learned manager gives solution of the issue/observation captured and also help them to choose best approach to sustain success for their project.

Lessons learned manager works with team members to discuss the matter regarding how to find factor affecting the issue? How to analyse any issue? To store the issue that could be available to any team member when they needed. Overall issues should be handled at lessons learned team, is under the guidance or leadership of lessons learned manager.

Lessons learned manager should decide the strategy for the matter like how to store lessons identified information? How to store lessons learned information? How it could be made accessible to all? How to share lessons learned information?

Lessons learned manager is answerable to Regional head of the organization. If organization is distributed across many geographical areas, then there should be lessons learned team at the every regional office.

### *4.13.1.2 Lessons Learned Analyst (1)*

Lessons learned analyst is playing important role in lessons learned team. The no of position for lessons learned analyst should be generally one. Depending upon the size of project no of position could be varying.

Main task allotted to lessons learned analyst is of analysis. When any issue arises or observation is captured, lessons learned analyst should analyse the issue. He uses so many techniques to analyse the issue and find out the factors affecting the issues to arise. He/she must analyse any observation very deeply.

Lessons learned analyst is working under lessons learned manager. He/she has to report lessons learned manager. Moreover, lessons learned analyst has to handle technical information writer as they are working under him/her. Lessons learned analyst used to collect information from observer or team members who face the problem or identify the problem.

He/she should also involve with the deciding the format for lessons learned document. The strategy to find the factors is decided by the lessons learned analyst. He/she should give the best solutions for any issue with the help of lessons learned manager. Though ultimately lessons learned manager should identify the best solution to sustain project success.

### *4.13.1.3 Lessons learned technical Information Writer (2)*

No of positions for the Lessons learned technical information should at least 2 positions. It could be vary depends upon size of the project.

Lessons learned technical information writer has to sum up all the information when lessons learned analyst is gathering information from the observer who capture the observation or issue or from the team member who faced the issue. All the information regarding any issue is listed or collated by lessons learned technical information writer. They have to arrange information regarding any issue in a proper format. This information is handed over to the lessons learned analyst.

Lessons learned technical information writer has to look after the matter like how to store lessons identified information? How to store lessons learned information? How it could be made accessible to all? How to share lessons learned information? All these things are handled by technical information writer.

Lessons learned technical information writer has to attend all the meeting held during the course of the project development process. They have to list out the issue raised during the meeting and prepare document for the same.

## 4.14 PARAMETERS OF LESSONS LEARNED TEAM

Lessons learned team should be very important organ of any organization. As the parameters of the lessons learned team is clearly depicted over here.

1) Accept the lessons learned team as part of project development process.
2) Lessons learned team should be given all the facilities which is providing to all the project development team members.
3) Lessons learned team should get an environment where team members can work freely and sincerely.
4) Lessons learned team should work in coordination with the other project team members.
5) Lessons learned team should provide latest technology to store and distribute the lessons learned.
6) Lessons learned team should not play blame game. That, they should not blame any personnel involved in project development for any issue raised.
7) Lessons learned team could get participation in of any of the project development team to analyse the identified issue or observation.
8) Lessons learned team has to attend the meeting held during the course of the project development.

9) Any of the project development team members who asked to cooperate in lessons learned should support the lessons learned process.

10) No one should take anything personal if lessons learned team member asked them for any issue raised.

11) They should be aware of the all the information regarding project development.

Above parameters should be set to get work done. To get benefit of lessons learned approach above parameters should be followed. So that, lessons learned team can work properly and sincerely to obtain correct output. This output can be utilized for any of the successor project development.

With this parameters lessons learned team can work smoothly and with ease. So, this is necessary to set such parameter for the lessons learned team.

## 4.15 FUNCTIONALITY OF THE LESSONS LEARNED TEAM

The type of work or functionality of lessons learned team should supposed to do is describe over here.

1) Lessons learned team should keep track of all the issues/observations raise during project development process whether the observation is of success or failure.

2) Lessons learned team should help other project team member in their need to solve any issue by finding out prior lessons learned document.

3) All the team members have to remain present to keep track of project development process.

4) Project team members can get help from lessons learned team at any stage of development.

5) Lessons learned team should work in such a way that project development process would be easy and flawless.

6) Lessons learned team should prepare templates for recording the lessons identified.

7) Lessons learned team should prepare template for recording the lessons learned.

8) Lessons learned team should store all the documents at central repository with latest technology.

9) Lesson learned team should make the provision to get the information of lessons learned with ease.

10) Lessons learned team is responsible to distribution of information through e-mail, blog, forum, etc.

## 4.16 ADVANTAGES OF LESSONS LEARNED TEAM

Following are the advantages which could be achieved by having the lessons learned team in an organization.

1) Lessons learned team help project development team in any hurdles, need or emergency.

2) By helping project development team lessons learned team would make development task stress free.

3) Project development task is very stressful and project development team remains in tension throughout the project development process. As lessons learned team makes this task stress free, project development team feel relax.

4) As project development team members feel relax, the efficiency of working of development team will go high.

5) In stress free environment development team would give better performance.

6) Better performance of the development team would lead to the success of the project.

7) Ultimately, it would lead to the success of the organization.

8) Learning from prior mistake would give benefit to the organization in terms of time saving. As from prior mistake or success development team can choose the best way of success.

9) It would also give benefit to the organization in terms of cost saving by avoiding the wrong way to be chosen.

10) It would be the best management tool as basis on prior lessons learned; better management plan would be formulated.

So, above given benefits would be achieved by having lessons learned team in any project development organization.

## 4.17 LESSONS LEARNED TEAM TRAINING

There are lots of issues arise during the development of the project. To cope with the newly raised issues lessons learned team has to develop themselves so they can manage the issues.

There should be training once or twice in year to train the lessons learned team. They should be aware about the new issues raised during the project development process. Training should be given to make them aware and how to handle the issues, the knowledge of this also imparted in the training course. There would be multiple solutions for the issues or observation captured and how to choose the best one, the knowledge of this should also be given in the training course.

Today's world is fast moving world with technology. Day to day lots of tools and technologies are introduced in market. So, to get awareness about new tools and technology that how to use them and what are the utilities of that should also team member can learn from this kind of training programs.

Lessons learned team should keep themselves up-to-date with latest technology.

### 4.18 LESSONS LEARNING PROCESS

Lessons learning process should consist of four phases. It is complete process from identification of problem to distribution of that event. The four phases are described below:

1) Analysis Phase: Problem identification should be done at this phase.

2) Remedial Action Phase: Identified problem should be getting corrected in this phase.

3) Storage phase: Lessons learned are stored with proper tool at proper place with proper format in this phase.

4) Distribution phase: Lessons learned are available for personnel or stakeholders who need it with entire institutes.

Following is the pictorial representation of Lessons learned Process.

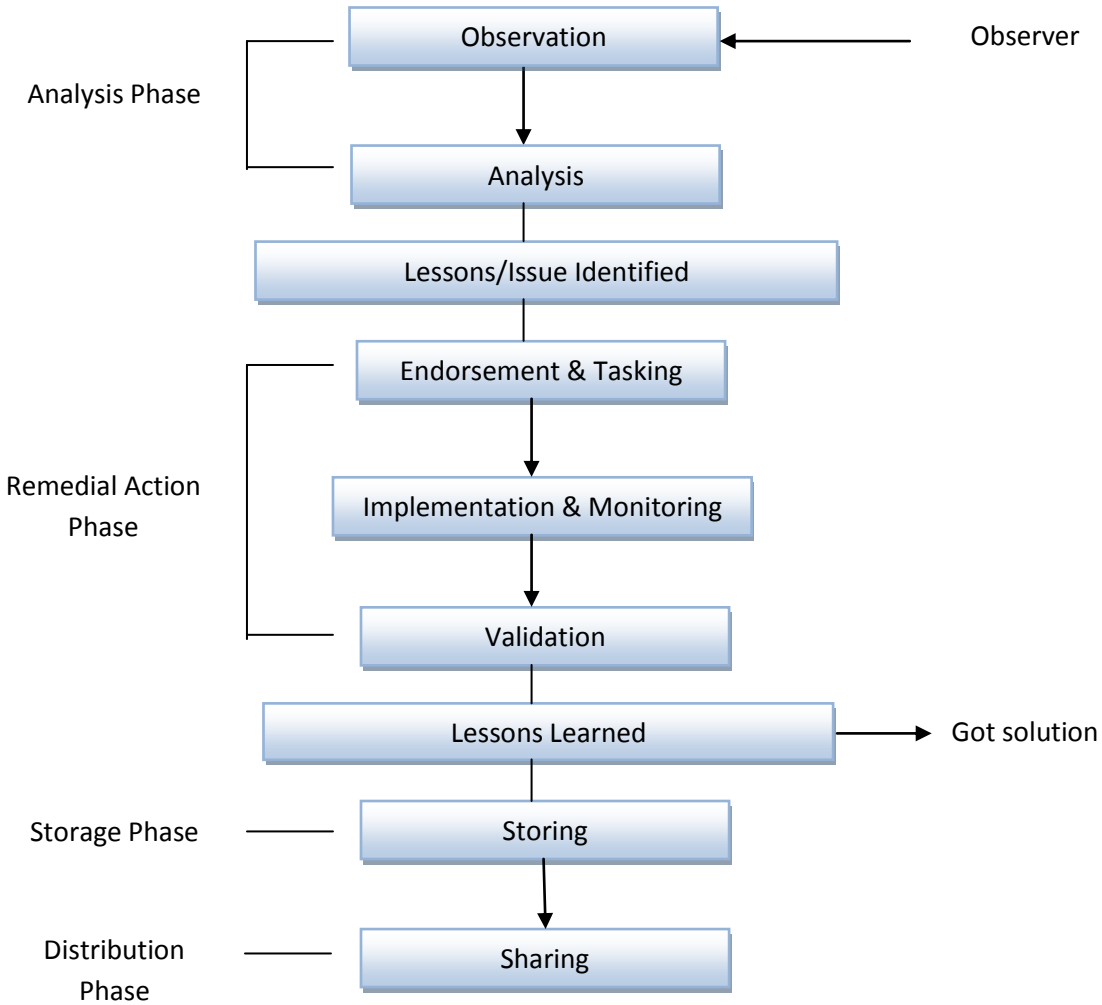



**FIGURE-20 LESSONS LEARNED PROCESS**

All the phase of Lessons learned process is described below:

### 4.18.1 Analysis Phase

As Researcher said above, problem identification should be done at this phase. This phase is consists of two different stages given below:

A. Gathering observation

B. Analysis

During project development, for a given activity, an expected outcome is desired. If expectations are either not met or exceeded, there is something to learn. Any differences from expected outcome should be documented as an observation. [42]Observation means “a comment based on something someone has heard, seen or noticed that has been identified and documented as an issue for improvement or a potential best practice.”

Analysis phase is start with gathering observation. Observation describes the sequence of events, condition under which the events occurred, and other related details. The responsible person who is doing the activity for which outcome is not as expected, should conduct some analysis on the factors to check why the activity differed from expectation and identify a proposed solution.

So, the analysis means “the study of a whole event by thoroughly examining its affecting factors.” Analysis allows discovery of the root cause of the observed problem or success. Once the root cause is understood, an appropriate Remedial Action that will address the root cause can be identified to correct the problem or get success.

[42] http://www.jallc.nato.int/newsmedia/docs/lessons_learned_handbook_2nd_edition.pdf

### *4.18.1.1 Gathering Observation*

Thoroughly understand about observation that an observation is the basic building block of a lessons learned process. The observation must convey the basic details of the issue, which are sufficient for further analysis. The observation must address the questions “what happened?” and “how did the outcome differ from what was expected?” Lessons learned Analyst should gather the observations as they occur. Once gathered, observation should be reviewed if the minor problem is there which can be neglected or analysis should be done to overcome from that problem. How observations should be gathered is described below:

To gather observations or factors affecting the observation should be in defined procedure.

- Capture Observation
- Manage Observation
- Tools available Manage Observations

- **Capturing Observation**

All the team members must have to understand that they have a responsibility to document observed problems and successes. All should actively participate in capturing observation. Any team member who is doing any activity, find any activity who does not meet the expectation, must be observed and it should be told to Lessons learned manager.

After getting acknowledge about observation or event lessons learned manager should identify how it is occurred? And what are the factors affecting the event to happen? Why this is happen? After finding answers of all this questions, lessons learned manager must have to discuss with the team member. The team member could be any person involve with the project, may be involved with the captured issue or may be not involved. Pointing out the issue can be done by the personnel not involve with the same project also. It could be any person of the organization.

Following is the format in which the information should be captured and its description.

| Title | Observer | Issue Type | Issue Category | Observation | Discussion of Root cause |
|---|---|---|---|---|---|
| Title of the observation | Name of the observer | Which contains the type of the issue like financial, technical, development, management, etc | Which phase issue belongs to. | What is the exact issue is describe here | Discussion of the root cause due to which the issue is arised. |

**TABLE-5 FORMAT FOR GATHERING OBSERVATION**

**Title:** This is the title given to the observation.

**Observer:** Depicts the name of the observer who would come across with the issue or observation.

**Issue Type:** This contains the type of the issue like financial, technical, development, management, etc.

**Issue Category:** This contains the information about the phase to which to issue belongs to.

**Observation:** This contains about the exact information of the observation/issue.

**Discussion of Root Cause:** This contains the discussion about the factors due to which the issue is occurred.

- **Managing Observation**

Observation will arrive from many sources of the organization. All the observation must be reported in proper format. The format include like what is the issue? What it the project name? What is the expected outcome? And what is the actual outcome of the event occurred now? Who reported the issue? Which phase of the development the issue concern? The date and time of the observation/issue reported?

After arranging them in a format, all the issue should be review by Lessons learned manager. If any raised problem does not affect the efficiency of the project development or the issue is minor then it should go off. If the issue is major then it must have attend to solve it.

After reviewing the observation Lessons learned manager will decide to include that issue or observation to the lesson's identified category.

To include the observation or issue to the lessons identified category following checklist must be observed:

- Is this an objective observation and not just an obvious complaint about something or somebody?
- Is this a problem with the system and not just a simple mistake by somebody?
- Does this adequately and correctly describe the observed situation?
- Would you spend your own money to fix this issue?
- Would you spend your own time fixing this issue?

If all the questions answer is yes then that observation/ issue must include in the lessons identified category otherwise consider that issue as minor issue and let it go.

- **Tool available to Manage Observation**

A tool should be used to support the collection of observation to ensure that observation can gathered, processed, prioritized, and shared. The tool which is used should be as simple as possible and should complement the organization's procedure for processing and sharing information. It can be collected in hard copy. But software solution offer best alternatives that can make tracking, processing and sharing observation easier.

When considering software tools to support observation collection, the following questions should be addressed, in addition to the usual cost and maintenance considerations:

- Is the software easy to use and familiar to users?

- How will the observation collection capability be deployed: stand-alone PCs; over a local area network; over a wide-area network; over the Internet?
- Will files need to be centrally accessed or circulated?
- Will the information remain current? How will versions be controlled?
- What contributing and editing rights and limitations are required?
- What browsing capabilities are needed?
- Can searches be performed readily?
- Can the information be updated easily?
- Can supporting information such as images be attached?
- What report generation capability is needed?
- What staffing processes will need to be supported by the software?

Above are the factors to choose the tool to capture and manage observation or issue.

Some tools are explained below:

**Web-based System**

Microsoft SharePoint Server and similar systems are web-based content manager systems. The use of SharePoint, when and where available, is probably the best way to collect observations because it has web parts which make it easy to create a simple form for collecting observations that can be used over an internal network or the Internet. Observers can simply click on a link, enter the data, and submit the form. Observation records will be automatically time stamped and tagged with the submitter's log-in name. SharePoint can automatically export the submitted observation to Microsoft Excel or Microsoft Access for further processing.

**Microsoft Office Software**

Microsoft Word, Excel and Access are simple and widely available tools that can be used to store and manage observations. Many users will be familiar

with these tools and will have the software installed on their computer, encouraging easy sharing. Ease of use and familiarity are important considerations in encouraging people to submit observations. A short overview of the advantages and disadvantages Word, Excel and Access have for supporting observation capture follows:

**Microsoft Word**

**Advantages**: Familiarity; ease of use; ease of setting up; ability to store metadata in file properties; ease of sharing.

**Disadvantages**: Difficult to manage many observations; no filtering of observations; limited sorting of observations; limited search capabilities; poor data integrity protection.

**Microsoft Excel**

**Advantages**: Familiarity; ease of use; ease of setting up; easy metadata tagging; powerful filtering, sorting tools; good search capabilities; easy to share.

**Disadvantages**: Merging independent data files is difficult; only 1024 characters display in a cell (in versions of Excel prior to Excel 2007); relatively poor data integrity protection (easy to delete and edit entries by accident).

**Microsoft Access**

**Advantages:** A relational database can store lots of data very efficiently; excellent browsing, filtering, sorting and custom reporting capabilities; good data integrity protection.

**Disadvantages:** A relational database can be very complex to set up and maintain; majority of users will be less familiar with Microsoft Access than with other Microsoft Office applications; Microsoft Access is not a component of some Microsoft Office installations; potential problems with publishing Access database files to document handling systems.

**Customized Software**

Customized software also can be used to record the lessons identified during course of project development.

### *4.18.1.2 Analysis*

Once an observation is considered to include in Lessons learned process, the next stage is its transition from an observation to Lessons identified. Analysis is generally has two phases: first to find root cause and second to determine Remedial Action for the observation. In this section, analysis technique is given to analyze the observation more closely. This analysis is done by Lessons learned Analyst.

To get into the technique first it must understand about what to analyze during analysis. Whatever the observations are collected, it must have to check that do they require further analysis or not. If they don't require further analysis then action plan is prepared for that observation and if it require further analysis then using some technique analysis should be done. So, to check that the observations require further analysis or not, the checklist is given below:

- Does the observation contain any causes of the observed issue (i.e. explanations of why the issue occurred)?
- Do the explanations of the causes (i.e. why it happened) seem to be correct?
- Are there no other immediately obvious explanations of why the issue occurred?
- Does the observation contain a recommendation (solution) that would address the suggested cause of the observed issue?
- Are there no other immediately obvious possible solutions to address the cause?
- Does the recommendation suggest an action body?

If the answer to all the questions in the checklist is yes, then the observation can be considered mature and no further analysis of the issue may be needed. Once the recommendation or activities are written up for

improvement or to correct the identified issue as a Remedial Action[43] and a suitable Action Body[44] is identified, who can execute the Remedial Action plan. Action plan is "the written plan of action and milestones developed by an action body to implement assigned remedial action for a lesson identified." Now this observation can be considered an LI. This Lessons Identified is recorded in a particular format, which is as below:

| Title of the Observation | Observation | Analysis | Conclusion | Recommendation |
|---|---|---|---|---|
| Title of the Observation | Note that the observation again describes exactly what happened. | The analysis provides a bit of a story and indicates how the conclusion was obtained. It is logical to follow. | Does not repeat the observation. It describes the overall observation, the cause of the issue. | Activities is to be done after conclusion is given. |

**TABLE-6 LESSONS IDENTIFIED TEMPLATE**

Following is the brief description of the Lessons Identified Template.

1) Title of the Observation

The title should be brief but specific. It should give a reasonable indication as to content of the observation.

2) Observation

A statement to describe what happened and how that different from expectations. This statement can be positive for example something that was observed to work better that expected or a work around or negative for example something happened that should not have or something did not happen that should have. It should be not lengthy.

[43] Remedial Action-An activity or set of activities that corrects an issue identified for improvement or facilitates the implementation of a best practice.

[44] Action Body- The organization or staffs who have to execute remedial action in association with a lesson identified. The action body develops an action plan to guide the remedial action activities. Action Body could be vary depending upon the issue belongs to which phase.

Observations should be restricted to single issues. Multiple issues should be divided into separate observations and cross reference to each other in the discussion section.

3) Analysis

The analysis explains how and why the observed issue differed from expectations. Reasons for success or failure and the circumstances surrounding or factors affected, are discussed. The analysis more elaborates the observations statement and answers the questions about the observation. It should explore all the factors, the analysis of the observed issue. It should include the history of the issue, the factors and environment, and any actions taken to work around the problem should be explained in detail. Then also result is negative, it should also explain in details.

But don't repeat the observation. Analysis should be to the point but include all data/information necessary to explain the issue and useful for further analysis.

4) Conclusion

The conclusion is a summary statement of the lesson that has been learned from the experience and the investigation into the root causes of the issues described in the observation and discussion. It must derive from the information contained in the observation and discussion.

In this section, don't go in much detail, and make sure that the conclusion stick to the observation and don't contain new information. No recommendation should be given in the conclusion detail. What have learnt from the observation and discussion should be included in this section.

5) Recommendation

The recommendation should be the suggested remedial action providing explicit advice on what must be done to repeat the success or to avoid and/or solve the problem. Identify exactly what needs to change in procedures, procurement of new equipment, change of the process structure, improved training or some new activities to be done. The recommendation should propose an action body to execute the new suggestion. The recommendation

should follow conclusion so that a proper guidance can be given and get the benefit of learning.

In this section don't contain the observation or conclusion. Recommendation should follow directly from the conclusion

| Title of the Observation | Observation | Analysis | Conclusion | Recommendation |
|---|---|---|---|---|
| Meeting are too time consuming at the later stage of the project development. | As half of the time is passed in discussing the matter which already discussed or known by team members | As the project reach its peak point, all the team members know everything about the project. As weekly meeting arrage, all have to remain present though they know everything. And discussion is also about status of the project which also known to the team member. If no of meetings are reduced from week to month then saved time of team members can be used in some creative work. Suppose if a weeking meeting take 1 hr. 1/2 hr is used to discuss known matters and 1/2 is utilize so, by doing monthly meeting of 1& 1/2 will be enough at later stage of development. so approximately 2 or 2 &1/2 will be saved. | If these no. of meeting are reduced then team member can save their time of meeting as well as transit time is also saved. They can utilize this time in doing other work. | From the observation and discussion, no of meeting should be reduced to once in a month. Make sure all the matter should be discuss in monthly meeting. |

**TABLE-7 EXAMPLE OF LESSONS IDENTIFIED**

However, if the answer to any of these questions is no, then the observation is considered to be raw and further analysis of the observed issue is required.

The analysis does not necessarily need to be carried out by professional analysts, but this is done by Lessons Learned Analyst who is the member of

lessons learned team. He/she has to analyse the observation or issue objectively and analytically to identify the root causes of the problem.

Issues or observations which are captured can be of any development phase of the project development. It can be of any resources or it can be of personal issue of the team members. So the issues can be of anything related to project development process or project development environment.

To analyse an issue Inductive approach is used. It is bottom up approach. In this approach first observation should be gathered. After getting observation fact (should be gathered in terms of the factors), due to which the issue is raised analysis should be done. After then issues are analysed to find patterns or trends. It means to find what kind of exact factors affecting the issues are there among the factors found during the analysis technique? There are some of analysis techniques are given below that is interview technique, Questionnaire technique, etc.

If same factors are there behind same issue this would form a hypothesis and based on this theory would be developed which shows the exact factors affecting the issue.

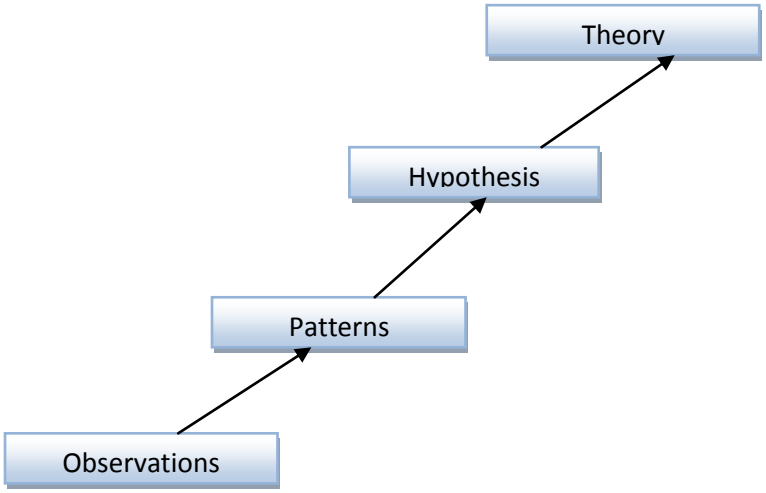



**FIGURE-21 INDUCTIVE ANALYSIS APPROACH**

The inductive approach may require more observation or more information about the observation before pattern can be found. For this reason, you may

require to collect additional data to facilitate your analysis. This could be facilitates by following analysis techniques.

**Analysis Techniques**

There are so many techniques to gather information. Some of them arranging interview, questionnaire, cause and effect, flowchart, etc. Let us discuss them one by one.

**Interview technique**

One way to collect additional information and issue related factors is to conduct interviews. To collect the additional information interview must be arrange with the observer or team member who faced the issue. Whatever the question answer would be there at interview, they must be recorded.

**Process of the interview:**

A common task for the Lessons learned analyst is likely to be gathering further information about observations in order to consider them to Lessons Identified. If there is time, this is best achieved through interviews with the person or people who submitted the original observation. Using the following process will give interviewee the best chance of getting all the information you need.

Interview is a process by which one can get knowledge of observation or issue. Find the basic purpose of the interview based on the original observation. Surrounding information of the observation or issue is also getting known by this interview. Following steps describes how to conduct or what to ask in the interview.

1) Introduce the basic idea

In this step explain about the main idea behind arranging the interview. Remind interviewee about the observation that the interview will be based on and listen what they ask.

2) Identify the exact issue/observation

After giving the basic idea about the interview, ask a number of questions to identify what learning came out of the observation. These should be ‘what’ questions like

- What were the key issues?
- What were the success factors?
- What worked well?
- What didn’t work well?
- What were the challenges and drawbacks?
- What would be the approach next time?

3) Explore root causes

If any of the steps find interesting then following question of series should be asked to more explore root causes.

- Why do you were successful?
- What did you put in place to ensure success?
- What was missing that caused that to happen?
- What makes you say that or reason to ask any matter?
- Can you explain how you achieved that?
- Can you tell me about that?

Such kind of questions should be asked to explore or to find more information about the any of the point of the previous step or find the exact reason of why such issue arise.

4) Ask about Advice

When root causes are clear then it must be asked about the advice of the interviewee, who has observed or faced the problem. Ask about the activities to be done to get best result.

- What would be your advice for someone else doing this in the future?
- If you were doing this again, what would you do differently next time?

- If you could go back in time and give yourself a message, what would it be?

5) Capture the notes and review it

During the interview, interviewer must have to note down the things. After completing the interview, he/she completes the notes for a while. He/she should ask for 1 or 2 days time to review it. If any corrections are there then making them correct the notes should be sent to the interviewee. Interviewee again checks it as the notes is same as he gave information or not. If any corrections are there then interviewee make corrections and ask them to rewrite it.

Above is the whole format for Interview session. Interview session contains the some sort of questions as like above.

**Questionnaire technique**

When a particular functional team is involved in issue/observation at that time questionnaires are used to collect additional data. When additional data is needed about an observation provided by an individual, questionnaire-style data collection is usually not as effective as interview.

Questionnaire can be prepared according to the observation and it is dependent on the development phase. Suppose any observation/issue regarding planning phase is there then the questionnaire would be like following:

- Was the project scope clear to the team members?
- Did the team members actively participate into the estimation process?
- Did the team members convey their opinion in planning phase?
- Did you state you opinion regarding the construction of the WBS?
- Did you state you opinion regarding procure management plan?
- Do you think newly possessed technology will be useful?

Such, so many types of questions can be placed at the questionnaire. But this should be dependent on the issue.

**Process for Questionnaire**

1) Identify the development phase of the issue/observation

First, identify the development phase of the observation. It means that the raised issue belongs to which phase of the development must be identified first.

2) Identify the issue

Then, get the complete information of the issue. Primary information must be got from the observer or the team member who face the issue.

3) Prepare Questionnaire

Once, issue and development phase is identified then questionnaire should be prepared. It must contain the question from which lessons learned manager can get more information for observation. Multiple copies of the questionnaire should be prepared.

4) Get the information

All the team members who are involve with the issue should get this questionnaire paper. They should give the answers regarding the questions. After they have completed the paper, it should be collected.

5) Capture Conclusion

This questionnaire should be analysed by the lessons learned analyst. After going through these papers, he/she reaches to a conclusion.

Then lessons learned team sit together and come to the final conclusion and prepare Remedial Action plan to be executed with the help of lessons learned manager.

**Cause and effect technique**

It is also very good technique to analyse the issue/observation.

- Focus attention on one specific issue.

  There are so many issues raise during course of development of the project. To find factors affecting the issue to be raised should be analysed by focusing attention on one specific issue.

Suppose during planning stage, there is problem in cost management plan due to man hours are wrongly calculated in schedule management plan. So, its affect the different management plan also like deliverable, work breakdown structure. So, there would be lots of connected issue. So, first have to identify the specific issue and focus on it.

– Organize and display graphically the various theories about what the root causes of an issue may be.

  Issues should be further investigated. Tentative root causes should be found out. They should be graphically depicted how they affect the issue to be raised.

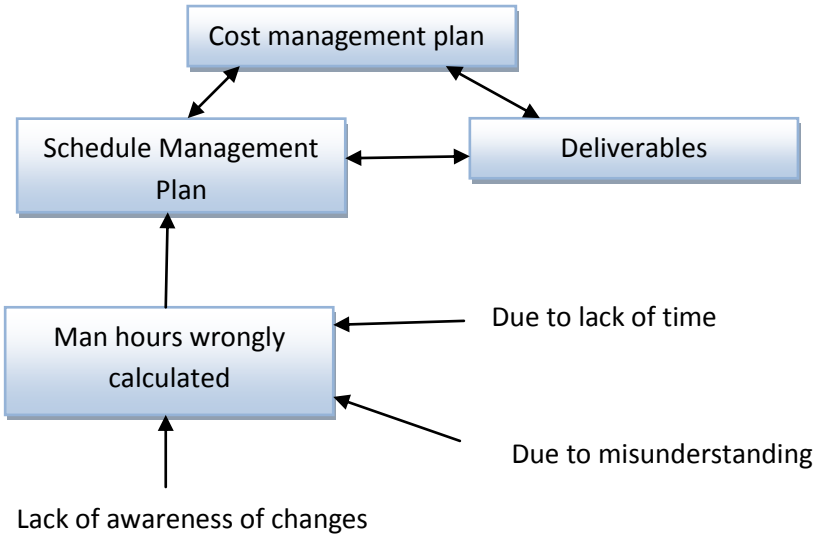



**FIGURE-22 TENTATIVE ISSUE**

– Show the relationship of various factors influencing an issue.

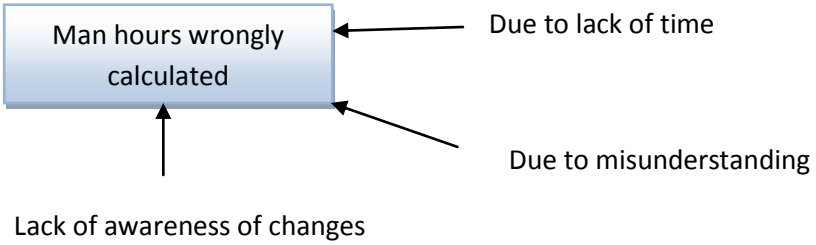



**FIGURE-23 INFLUENCING FACTORS**

- Provide additional insight into process behaviours.

  Due to lack of time or in hurry, if the Schedule Management plan is prepared, the chances of causing error would be very high. So one this could be the root cause. If during planning discussion, misunderstanding would be there which would impart the cause of issue or observation. If technical information writer does not understand what is the exact matter then also such kind or error would occur. If any changes in man hours if would be known to the technical information writer properly otherwise such kind of error would be occur.

  This issue would affect lots of plans and when any of the plans will execute then the issues will be raised.

- Focus the analysis on the causes, not the effects or symptoms.

  So, during analysis focus should be there to find the root causes but not on the effects of the cause. When causes are identified then the focus should be there to plan remedial action for the betterment of the issue which was raise.

**Flowchart technique**

Flowcharts are used to represent a process broken down into less complicated sub-processes. By describing only a limited number of steps or activities at any one stage, the overall process becomes more manageable and understandable. Cross-functional flowcharts are used to illustrate which part of an organization performs particular activities or functions, and are useful in understanding organizational relationships.

In above example all the planning process should be depicted through flowchart and how the factors affect the planning process should also depict in the flowchart. By doing so, root causes could be identified and then according to them remedial action plan is created.

So, there are lots of technique through which root causes can be identified and then the issues should be identified and decides whether it should be included in lessons identified or not.

The aim of the analysis step is to provide an explanation of why the issue described by the observation occurred—i.e. the root cause—and provide a solution to fix the issue. After the analysis, the lessons identified template can be completed to record the resultant Lessons identified.

### 4.18.2 Remedial Action Phase

Once a lessons/Issue identified has been generated with a suitable proposed remedial action and associated action body, effort should focus on making it into a lessons learned. Transformation of lessons identified to lessons learned depends upon everyone involved leaders, stakeholders, and lessons learned practitioners and lessons learned team who has to follow basic principles of cooperation, coordination and communication.

Cooperation should be got from those personnel or team members who are involved in learning a lesson either full time or assist in achieving progress. The team member may be involved completely or partially like he/she can be observer or the person who faced the problem. However team member is involved they should have to cooperate to the action body to execute remedial action by preparing action plan.

Coordination must be there within the team members of the action body to plan their efforts effectively to ensure a timely and effective output of their work. If coordination is not there again chances of occurring mistakes would be there. Due to lack of coordination new issues would be raised.

Communication will facilitate cooperation and coordination between team members. Ensure all personnel are informed about the plan and about the success of the remedial action. It will inspire others to participate in lessons learned process.

#### *4.18.2.1 Process how Action plan formulated*

1) After identified the issued to solve and analyze them to find factors causing issue; remedial action should be given by lessons learned team. This remedial action is the solution of the raised issue.

2) To implement this remedial action first it should be approved from highest authority. This highest authority should choose the action body to implement remedial action.

3) Action body should prepare action plan to execute remedial action. This action plan should describe how to execute the remedial action to solve the raised issue.

4) Remedial action plan is then executed by action body.

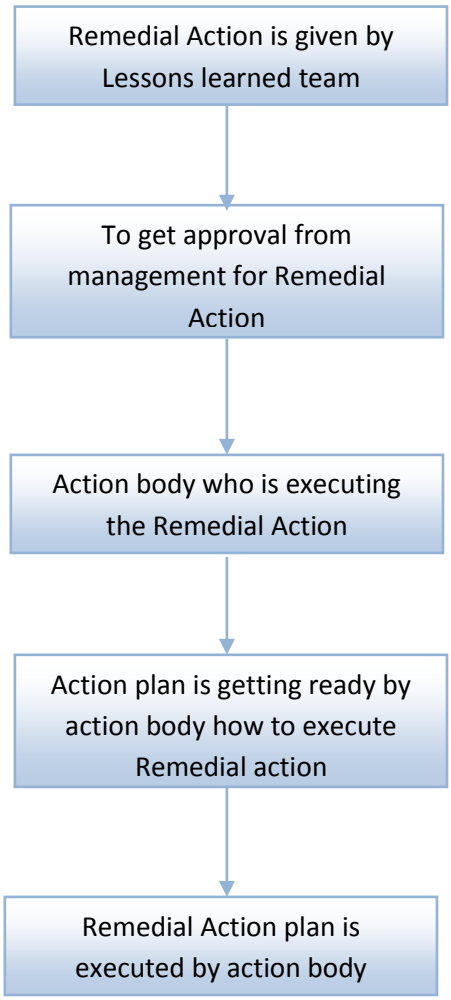



**FIGURE-24 PREPARING REMEDIAL ACTION PLAN**

After preparing Action plan following step should include in Remedial Action phase to execute the Remedial Action plan. By going through these steps Lessons Identified now converted to Lessons Learned.

A) Endorsement and Tasking

B) Implementation and Monitoring

C) Validation

All the steps are described follows:

**A) Endorsement and Tasking**

The endorsement and tasking of an Action body to complete the Remedial Action is the first step in turning a Lessons/Issue Indentified into a Lessons Learned.

**Endorsement:** It means the remedial action plan prepared by lessons learned team should be approved by the authorized committee. Authorized committee can be change control board (CCB)[45] or management group. This implies a process of review that includes checklists to check completeness and accuracy of the remedial action with respect to root causes and implementation of recommended remedial action.

Checklist is given below. Remedial action should contain all the information which is needed for approval.

- Is the remedial action is worth to execute get success in business?
- Is the remedial action is worthwhile spending valuable time, effort and money to achieve success through executing the plan?
- Is the remedial action is executed by experienced person and he/she will manage the action body?

[45] Change Control Board is the committee who approve if any changes needed in management plan during software development.

- What resources are required to carry out the work that is needed to complete the remedial action?
- Where the resources come from?
- How will action body achieve the necessary level of quality for the project?
- What requirements should the action body meet to prove the quality of the solution of raised issue is good?
- What are the deliverables at different milestones?
- Would the change be worth for getting success in raised issue?
- Is the remedial action plan is of good quality than original plan?

Depend upon the type of issue checklist should be prepared. Remedial action should analyze through this checklist. Approval of the remedial action is depending on this checklist. If it covers all the answers then authorized committee thinks it is worth to implement then they give approval.

**Tasking:** Action body is selected to implement remedial action. It means selected action body will be responsible for implementing the remedial action. Action body should prepare remedial action plan. Certain authority is provided to action body to execute the remedial action plan. The level of authority required will vary depending on the authority needed to execute the remedial action plan.

The lessons learned manager should support the endorsement and tasking by preparing lessons identified for presentation to the decision makers and to coordinate and administer meetings where endorsement and tasking takes place.

### B) Implementation and Monitoring

Once endorsement and tasking is complete, it is time for implementation and monitoring of the Remedial Action. The Action Body tasked with the Remedial Action should develop a Remedial Action plan for implementation. To assist in the monitoring of the Remedial Action plan, a small number of significant

milestones should be defined. The leadership should monitor these key milestones to measure success of the Remedial Action plan implementation.

The lessons learned manager's role in supporting the implementation and monitoring will usually focus on monitoring. Management should review for status of all Remedial Actions being implemented and must be informed of any cost, schedule, or management risks to implementation.

**C) Validation**

Validation in the lessons learned process is the act of ensuring the completed Remedial Action has correctly and completely solve the original issue observed. The process and level of effort required to validate will be determined on a case-by-case basis.

Validation means Impact of the Remedial Action. Here, effect of remedial action is find out. Is it work as planned and result is arriving as per requirement or proposed in remedial action plan? Such things should be observed to measure the progress.

There can be many risks which affect the implementation of Remedial Action Plan. Those risks can delay or halt completion of the remedial action. These risks could be lack of quality staffing, lack of adequate resources, and lack of adequate training for staff involved in the process. Someone must be there to take leadership and guide them from management team to reduce these risks.

Lessons learned manager can help the management team member in mitigate any hurdles or risk.

When remedial action has produced desired result then now issue is solved. At that time remedial action phase is completed. Now lessons learned document should be prepared.

***4.18.2.2 Lessons learned document*** should contain following information:

**Title:** It should contain title of the observation or issue raised.

**Date:** It should contain date on which it is captured.

**Project:** It should contain name of the project to which the issue belongs.

**Phase:** It should contain the project development phase from which the issue is noticed.

**Place:** It should contain the information about the place where issue raised like the address of the office.

**Location:** It should contain the information about the location or region if it is multinational organization.

**Submitter or observer:** It should contain all the information about the person like name, designation, belongs to which department, etc. who identified the observation or issue. That person can be a team member of the ongoing project or can be personnel outside of the project team.

**Person of the Team (POT):** It should contain the all the information about the person to whom submitter or observer has talked about the issue first time.

**Observation:** It should contain the exactly what has happened? It defines the observation.

**Expected Result:** This field should contain the discussion about the expected result or outcome, which is not achieved by the completed activity.

**Achieved Result:** This should contain the information about the output which is received by activities done.

**Discussion or Analysis:** This section of lessons learned document should contain the information about the analysis of issue raise. Why this output received and which factors are there to affect the current result to achieve.

**Remedial Action:** It should contain the information about remedial action to be taken to solve the issue. It describes what to do solve the issue.

**Action Body:** It should contain the information about the action body that is going to execute the remedial action. Information should contain name, designation, etc. information about the personnel in action body.

**Remedial Action Plan:** It should contain the information about the remedial action plan. It should contain all the information about how to execute the plan.

**Impact of Remedial Action Plan:** It should contain the information about the impact of remedial action plan. It contains the information of the effect of the remedial action plan, whether it is of positive and negative.

**Result:** It should contain the information about the result after implementing the remedial action plan. Describe the positive result or negative result whichever is achieved in all aspects of execution.

**Conclusion:** It should contain the information in which scenario the given remedial action should be implemented.

**Reference:** it contains the reference from where the remedial actions got.

**Status:** It should contain that remedial action works or not.

All above information should be there within the lessons learned document. Lessons learned document can be like:

After finishing the remedial action phase lessons identified are now turned to lessons learned. Now it must be stored with some tool to facilitate the ancestors for using them for forthcoming projects.

### 4.18.3 Storage Phase

Throughout the lessons learned process, to track the progress documentation is needed. When a lessons learned is achieved, it should be recorded in supporting information and documentation; otherwise this lesson learned is of no use. If it is not captured in proper format to document then forthcoming project will not get benefit of lessons learned of ongoing project.

#### *4.18.3.1 Tools to support lessons learned document*

Organizations create or adapt various tools for this purpose. There can be customized software to capture the lessons learned or some software used like spreadsheets or database can be used to capture lessons learned.

Spreadsheets can be used to store lessons learned data. The advantage of using a spreadsheet is that one can see all of the data in one shot. Spreadsheets can also filter data very simply using drop down menus.

The advantage of using databases is that they generally have excellent form generation and reporting capabilities, and are relatively used to customize and program. A common difficulty in using a database is that as the person that developed it leaves the organization, it will be of no use further due to unavailable of supporting documentation. So, if database is used then proper documentation should be prepared which will not disturb the information sharing approach in future.

Customized software can be used to store lessons learned document. So, these tools whether it is a spreadsheet, database or customized software, they can be useful to document the lessons learned.

Lessons learned document should be stored at central repository of the organization so that everybody can access the lessons learned document and get benefit from the document by learning and by implementing the idea in their project.

Lessons learned can be share by many ways, which is shown in next phase of lessons learned process.

### 4.18.4 Distribution Phase

The value of a Lessons Learned process is only realized when the information generated by the process is available to the people who need it particularly at time when they need it. Lessons learned information sharing generates organizational knowledge and leads to a permanent improvement in organizational performance.

However, there are great benefits in overcoming these concerns and sharing. Sharing knowledge generates better results in business.

Lessons learned team should make provision for the sharing of the lessons learned. Ultimately lesson learned team manager is responsible for information sharing. He/she must know:

- With whom the lessons learned information should be shared?
- When to share information and at what stage in the process it can be share? What lessons learned information to share?
- How to share lessons learned information?

### *4.18. 4.1Whom the lessons learned information should be shared with?*

In sharing a lessons learned it is not enough to simply publicize it. Some consideration must be given to who will benefit from the lessons, and this group is referred as target audience. Care should be taken when sharing lessons to ensure the lessons learned will be significant to the target audience and so, promote effective learning. Generally target audience is that who needs the information to improve his efficiency in project development process.

Generally there are two types so audience are there:

**Project Management Planner**

Project management planner needs this lessons learned information in planning the project management plan. They can design the project

management plan learning from the ancestors or previous project, which can be useful in their ongoing project. Planning by learning from previous project would help organization in doing mistake which earlier project has gone though. So, care must be taken and mistake made in previous project must be avoided.

**Training purpose**

Lessons learned information is needed to train the personnel in the organization. They can be aware about the issue which could be arising during the course of project development. They can learn from the information given in lessons learned that will improve the efficiency of the training people.

## *4.18. 4.2 When to share information?*

Sharing is not an event which should be occur a single in lessons learned process but it is something that to be started as early as possible and to be repeated frequently throughout the whole process.

Lessons learned information sharing can occur at any time during the lessons learned process, as it is depicted as continuous process throughout the software development. It is not distributed as the last step or as part of remedial action implementation. It involves sharing information but not formal information like action review of remedial action or intermediate report during ongoing process of lessons learned. All the Lessons learned information must be authenticated.

**Information must be secured.**

Lessons learned information should be managed with an emphasis on sharing, balanced by the considerations for security.

Sharing lessons learned, when not managed properly; result in accidental disclosure of classified information to someone who does not have to access the information. So, information must be classified and must be secured so

that risk of unauthorized disclosure is avoided. Sharing of information must be well-managed. Only authorized person can access the lessons learned information. In multinational units, security must be maintained. Lessons learned manager is responsible of secure sharing of information.

**Good and bad both issue should be shared**

When a good practice is noted, there is a natural desire to tell everybody about it immediately. This is possible, but should be done with care that until the practice has been analysed properly to determine the conditions and circumstances where it is valid, and how it can be work smoothly. The danger with sharing good practices too early is that people may assume this is enough to reach a Lesson Learned and might done mistake.

Equally, there is a natural wish to keep secret or minimize ineffective or unfavourable issue, or to blame negative outcomes on human error rather than ineffective plans, techniques, or procedures. So, Blame game should not be played as it down the moral of team member to accept the mistake. Lessons learned manager should encourage the personnel reporting of mistakes, by making the distinction between simple human error and more systematic problems. When an ineffective or negative practice is kept secret or minimized, it stops others the opportunity to learn from it, and it also restricts the chance to use knowledge and information or close to get gain through experience to improve.

**Quality of information must be maintained**

Lessons learned information must be maintained. It should not be the row information. It must be mature information which leads other to get benefit from this information. Before entering any issue or observation to the lessons learned repository it must have to gone through proper process. Proper analysis must have to be done of any issue or observation. It should be some concrete information which could make others to get benefit.

So, lessons learned would be of good quality with concrete information. By using this information one could solve their problem and could get success in their project development process.

### *4.18. 4.3 How to share Lesson learned information?*

Lessons learned information should be published by pushing and pulling of information. As name suggest pushing means approaching and pulling means fetching. Pushing information is the approach that new information is dynamically delivered to consumers or subscribers as it becomes available, while pulling information is the approach that consumers have to regularly check to see if new information has become available. An example of pushing information is the distribution of newsletters and the sending of e-mails to subscribers when something happens like the posting of new information on a portal. An example of pulling information is added information to a database where people are likely to go to find the information without being alerted that new or updated information has been published.

Most organizations will choose to use a combined push and pull approach. They sent the information to their subscriber and subscriber could also get the information from the organizations database.

There are lots of means for exchange of lessons learned information. Some of them are given below:

**Communities of Interest**

Communities of interest engage in software issue learning and are groups of people who have common goals interacting with each other. They gain knowledge by being part of such communities.

There are a number of benefits to participating in a community of interest, including:

- Solving problems.
- Developing new capabilities.
- Empower best practices.
- Standardizing practices.

- ‘Institutionalizing’ best practices.
- Time saving.
- Increasing skill sets.
- Avoiding mistakes.
- Creating new knowledge.

There are number of problems to participating in a community of interest, including the reliability and classification of information, technological problems such as connectivity, and rigid mindset against information sharing. Now a day personal internet connections considerably improves ability to share information. Personally knowing and trusting the individual from whom you are asking for information, or to whom you are sending information, helps information sharing. Sharing of lessons across national boundary, via social networks could be risky. There can be lack of security and chances of theft of information.

Many communities of interest are supported by forums that allow them to share their experiences. Participating in forums allows for strengthening personal contacts to encourage the sharing of information. Examples of forums include training events, conferences, working groups, etc., or virtual forums online, including discussion groups and blogs.

**Request for information**

Some organizations offer a request for information service, where individuals requiring information on a particular topic can make a request and the lessons learned team will search the lessons learned information they have in order to respond to the request. The ability to respond to the request will often depend on priority of the issue raised during course of project development or authorized person demand.

**Training**

Training events at the beginning of project development, at the beginning of development phase, or as part of ongoing processing are good opportunities to engage staff on the benefits, opportunities and requirements of a lessons learned process as well as to inform staff about the latest lessons from the field of project development.

**Information technology**

There are several information technology tools that support the sharing of lessons and information, including:

- Portals
- Databases
- Knowledge repositories such as Wikipedia
- Blogs.

A technology solution can provide easy access of many different types of lessons learned information. However, the information stored by information technology must be latest otherwise there is no use of sharing of information. So, information shared via information technology tools must be reviewed after some interval of time. That must keep up-to-date and latest information of lessons learned issue which provide latest guidance.

**Publication**

There are several ways to guarantee lessons learned information reaches those within and outside to the organization. Information should be regularly sent by:

- Newsletters.
- Reports.
- Booklets/Handbooks/Leaflets/Posters.
- Email Blasts.
- Blogs/Bulletin boards.

## 4.19 LESSONS LEARNED THROUGHOUT SDLC

Lessons learned approach should incorporate in SDLC model to improve the efficiency of the project.

Lessons learned is the learning gained from the process of performing the project or during course of development of the project. Generally, most of the company avoids keep tracking of lessons learned. Only some of the company does lessons learned activities and that is also not taking seriously. This is also at the time of project close-out. But this should not be done.

Lessons learning process should given a proper time as efficiency of the project depends on the lessons learned. So, lessons learned should be incorporated as “Development Phase” of any SDLC model. Though this activities is also useful in development of the project. It improves the efficiency of the project.

Moreover, lessons learned should not be the close out process only. It is not the activity which is only done at the end of the project but it should be a continuous process which should be done throughout the project. Lessons can be identified and documented at any point during the project development life cycle. The reason for this described earlier.

The purpose of documenting lessons learned is to share and use knowledge derived from experience to:

- Encourage the recurrence of desirable outcome
- Stop the recurrence of undesirable outcome

Lessons learned should draw on both positive experience – good idea that improve project efficiency or save money and negative experience – lessons learned only after an undesirable outcome that has already occurred. Every documented lesson learned should contain following information.

- All information of project and about the observer
- A clear statement of observation
- A background summary of how the lesson was learned
- Conclusion how the lessons may be used in the future

At any point during the project life cycle, the project team and key stakeholders may identify lessons. Then lessons are analyzed, formalized and stored during project development life cycle.

Upon project completion a lessons learned activities should given the time of some days. So, the stakeholders can analyze on identifying project success and project failure. They can include recommendation to improve future performance of projects.

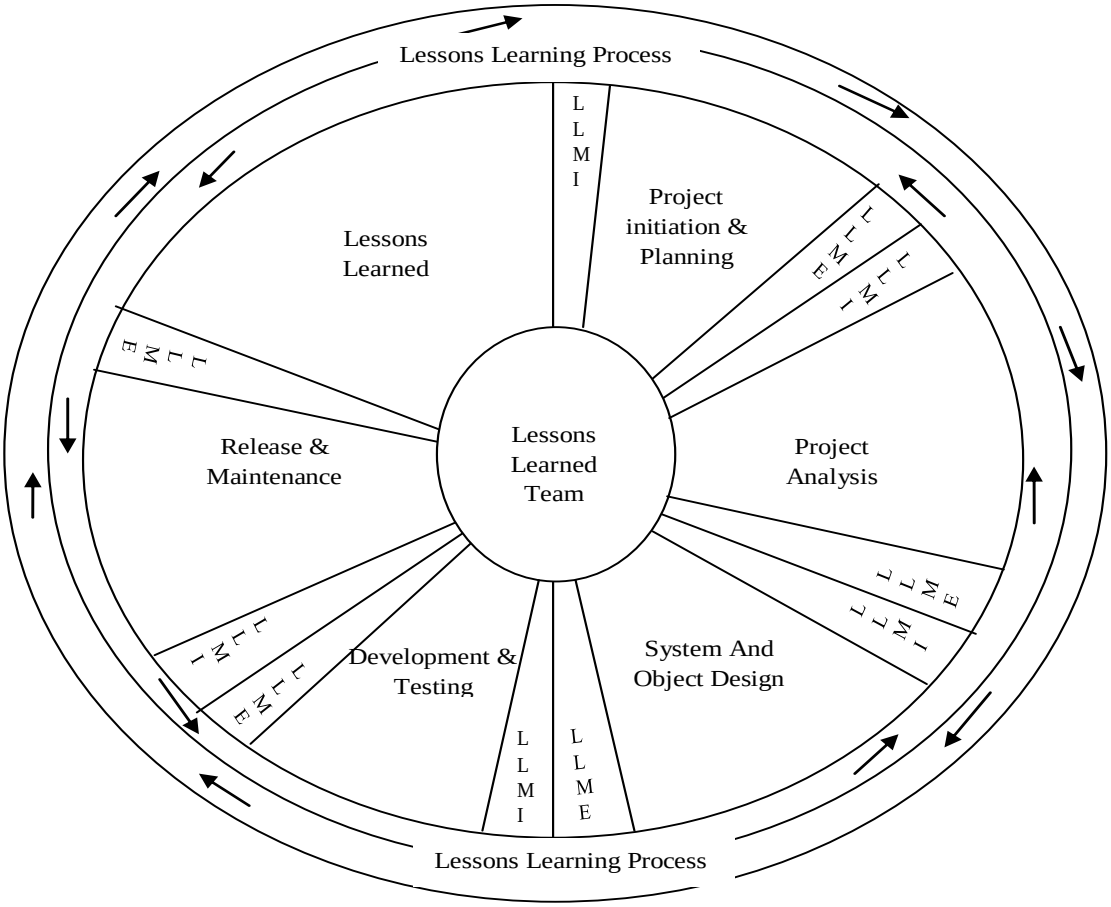
The ultimate purpose of documented lessons learned is to provide future project teams with information that can increase effectiveness and efficiency and to build on the experience that has been earned by each project during development period or at the time of end of the iteration of the project or at the end of the project.

When it disseminate properly lessons learned provide a powerful method of sharing ideas for improving work process, operation, quality, safety and cost effectiveness, etc. and helps improve management decision making and employee performance through every phase of a project. Lessons learned also helps to support some of the tougher times passed during the project's life and helps future project managers avoid similar difficulties.

## 4.20 LLMODEL

As above described, Lessons Learned Team must be there to look after the lessons learned affairs throughout software development process.

We know that any software which is developed is passing through many stages of development. Software development is quite tuff and lengthy job to do. Lots of problems come across during the development of the software. Software development is a continuous process during which problem related to development arises and development team has to solve it. The type of the problem or issue raised during the development of different software may be different or same. If the issue or problem is of same type and repeatedly occurring then there will be waste of time to invest the time in finding the solutions. There must be some permanent solution of those problems.




LLMI – Lessons Learned Meeting at Initial Phase

LLME – Lessons Learned Meeting at End Phase

**FIGURE-25 LLMODEL**

Lesson learned document serve this purpose. Lessons Learned is there to facilitate the software development process. Though lessons learned are not only record of problems or issue or failure but success event is also recorded in lessons learned document so, that successor project would get benefit from these records.

In above depicted figure, researcher has tried to incorporate lessons learned to SDLC as a “Lessons Learned Process” throughout SDLC. It is also incorporate as a phase in SDLC.

Lessons Learned is the process where problems are identified throughout the project development process and solutions are provided that can be useful to the successor project development. In this process issues which are noticed to be best is also recorded in lessons learned for the successor project development.

Lessons learned process is the process which should occur during the whole software development life cycle. Lessons learned process should attach with all the phases of the software development life cycle, as lessons could be gathered from any of the phases of the software development.

All the phases of the software development should start with the lessons learn meeting as depicted in the above figure. In above figure it is indicated by LLMI. In this meeting all team members like development team members, lessons team members, and technical staff should remain present. This is the meeting where prior happened/recorded lessons learned issues should be discussed. Here the discussion of the issues would be related to the phase. What would be the best practice to follow to develop software is discussed in this meeting. How care should be taken to tentative issues which might arise during the phase of the development. All these matters are discussed in this meeting.

Then during the ongoing phase of the development decision to mould the project should be taken on the basis of the lessons learned. Development process is proceeding further. Then after if any new issues arise during development for which solution is not there in lessons learned document, then

development team has to find the solution and record that issue for lessons learned.

The kind of issues or problems must be identified and recorded as described in the “Lessons Learned Process”.

At the end of the phase again lessons learn meeting should be arranged that is called LLME to discuss about newly raised lessons learned issue which are recorded in the “Lessons Identified Templates”. After discussion in the meeting, Lessons learned document should be prepared, which is known as “Lessons Learned Template”.

Then this experienced solve issues should store at common repository so that team members can utilize this experience in solving the problems of current ongoing project.

As depicted in above figure there should be meetings arrange in the starting of the any of the SDLC phase and at the end of any of the SDLC phase.

Below is the description of the meeting how they should be arranged.

### 4.20.1 Lessons Learned Meeting in Initial of the Phase

All the previously experienced issues related to the concern phase are discussed in this meeting. Time allotted to this meeting is of approximately 45 minutes.

#### *4.20.1.1 Purpose of the Lessons Learned Meeting*

Lessons learned topic described from previous projects of similar kind, which are already recorded discussed in this meeting. The discussed issues should be related to the phase only.

So, team members get aware of issues which might be raised during the phase.

### *4.20.1.2 Participants of the Lessons Learned Meeting*

All the stakeholders involved in the project can attend the meeting. Stakeholders can be project sponsor, development team and lessons learned team. If the observer is outside of the team then he/she should involve in the meeting.

### *4.20.1.3Facilitator of the Lessons Learned Meeting*

Meeting should be managed by experienced facilitator to accomplish the objective of the meeting within the prescribed time.

### *4.20.1.4 Lessons Learned Meeting*

**1) Introduction and Ground Rules (Approx 10 minutes)**

Review the original purpose, budget, timeline of the project.

**Ground Rules:**

- Cell phones and laptop should be put aside so that every stakeholder can be a part of the meeting.
- Project discussion is encouraged.
- Criticism of people is not permitted.
- Participation should not be destructive. It should inspire the development team.
- Meeting should help in learning from the previous issue.

**2) Discussion of Lessons Learned Issue: (Approx 30 minutes)**

- Issues related to the particular phase should be discussed.
- Cause of that issue should be discussed.
- Remedial action should also be discussed.

**3) Close out of Lessons Learned Meeting: (Approx 5 minutes)**

- Clarify if any doubt is there with the stakeholders.
- Encourage the development team to develop good quality project.

### 4.20.2 Lessons Learned Meeting in End of the Phase

All the newly raised issues are discussed in this meeting. Time allotted for this meeting is also of approximately 45 minutes.

#### 4.20.2.1 *Purpose of the Lessons Learned Meeting*

The observations are identified and remedial actions are taken to solve the raised issue during the ongoing phase. All these fact are recorded as "Lessons Identified" during the Phase. Now in this meeting they should be verified and then converted into "Lessons Learned".

#### *4.20.2.2 Participants of the Lessons Learned Meeting*

All the stakeholders involved in the project can attend the meeting. Stakeholders can be project sponsor, development team and lessons learned team.

#### *4.20.2.3 Facilitator of the Lessons Learned Meeting*

Meeting should be managed by experienced facilitator to accomplish the objective of the meeting within the prescribed time.

#### *4.20.2.4 Lessons Learned Meeting*

**1) Introduction and Ground Rules (Approx10 minutes)**

Review the original purpose, budget, timeline of the project.

**Ground Rules:**

- Cell phones and laptop should be put aside so that every stakeholder can be a part of the meeting.
- Project discussion is encouraged.
- Criticism of people is not permitted.
- Participation should not be destructive. It should inspire the development team.
- Meeting should help lessons learned team to develop proper document of the Lessons learned issues of the ongoing phase.

**2) Discussion of Lessons Learned Issue: (Approx 30 minutes)**

- Issues raised during SDLC to the particular phase should be discussed.
- Cause of the issues should be discussed which helps in formulate Lessons Learned document.
- Remedial actions should also be discussed which helps in formulate Lessons Learned document.

**3) Close out of Lessons Learned Meeting: (Approx 5 minutes)**

- Clarify if any doubt is there with the stakeholders.
- Encourage the development team to develop good quality project.

### 4.20.3 Lessons Learned Phase

This is the phase where the overall project should be analyzed. What gone well and what gone right for the project should be analyzed. Lessons Learned documents are prepared here and should be stored at global repository. Then after these lessons learned information should be made available to all the employee of the organization and also provision should be made to make it available to all the people engaged to software development profession. How lessons learn process work is described earlier.

Overall project analysis should be made here. Some of the facts related the project is collected here like.

- What happened in this project?
- Where were we beyond schedule?
- Where were we on schedule?
- Was the time allotted to the every phase adequate?
- Did we correct define the goals for the project?
- Was the cost is beyond the cost estimation?
- How we personally feel after this project?

- What did we learn?
- Were all the parameters set by us gone well?

Above like questions should be discussed and find the answer. They should be recorded in the proper format as discussed above and should be stored.

### 4.20.4 Parallel Utilization of the Lessons Learned

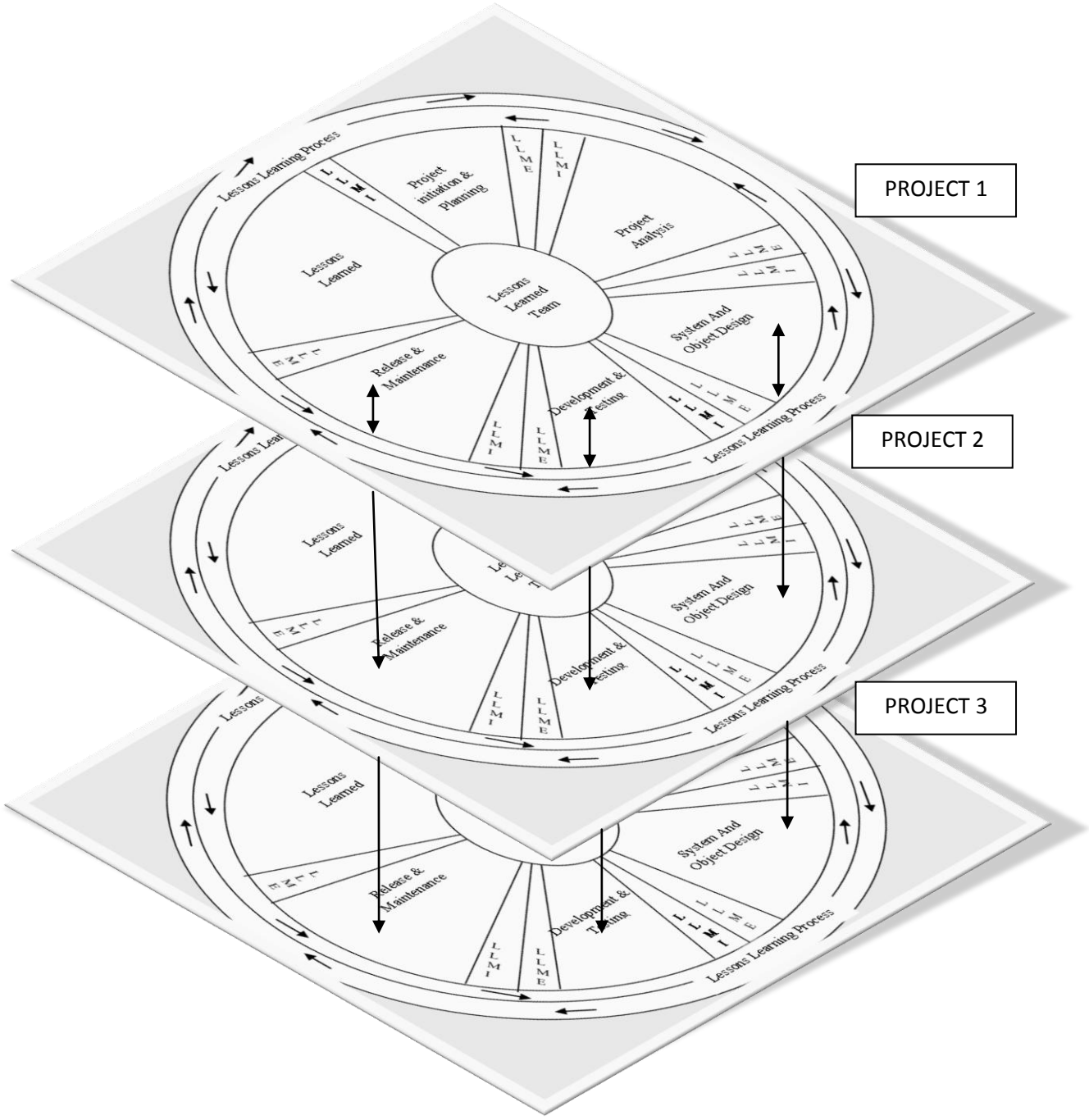
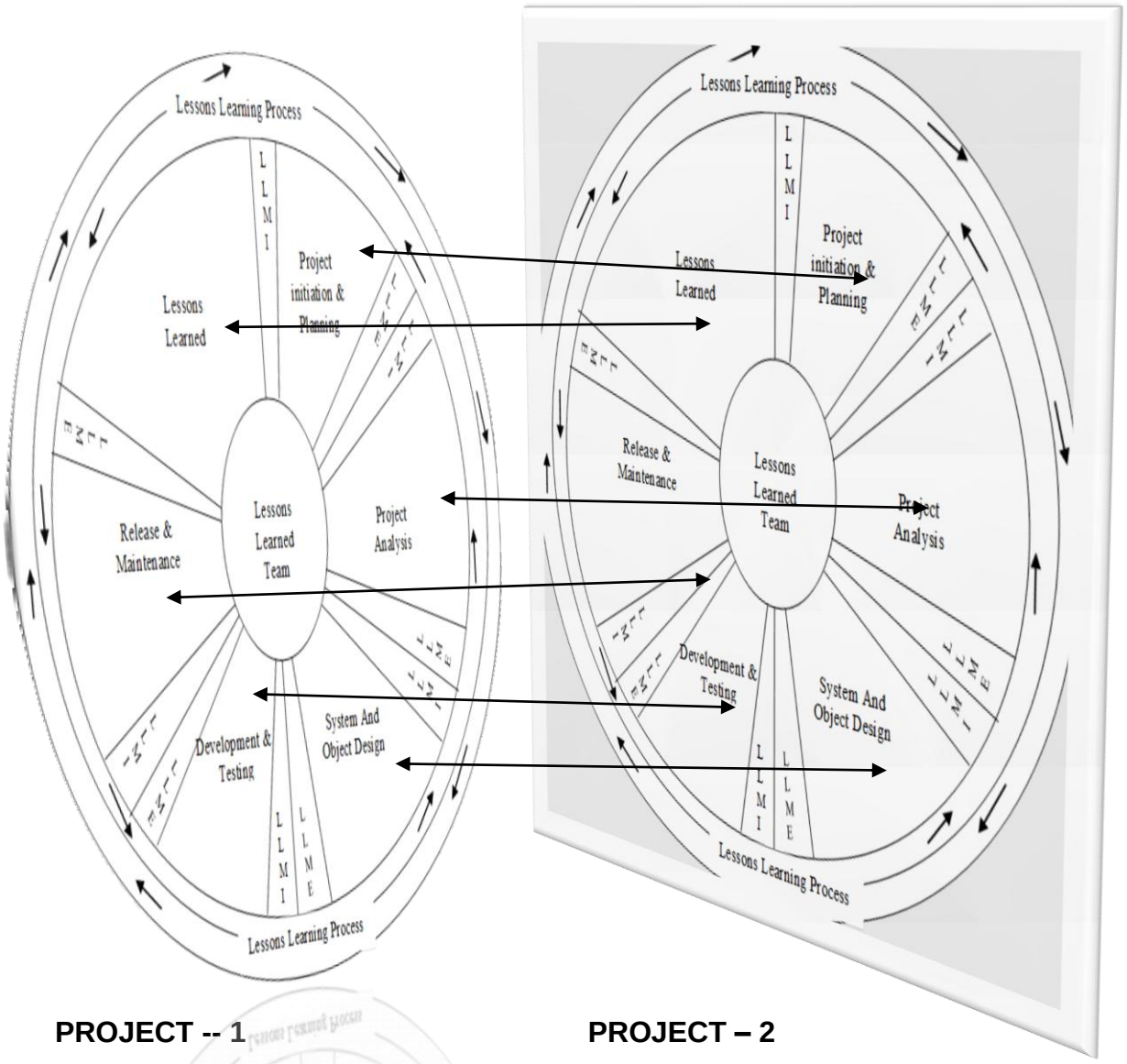



**FIGURE-26 PARALLEL UTILIZATION OF LESSONS LEARNED**

Above figure depicts the inter utilization of lessons learned. Lessons learned can be utilized parallel as above figure show that, three projects of same kind would ongoing parallels. As researcher discuss that lessons learned is continuous process during the time of development of any project, then lessons learned document prepared in one project can be utilized by another project also.

All the documents prepared during the any phase of the development can be utilized by another project which is ongoing same time. By doing so, management team can take better decision to develop the project fast as to complete in decided time with predefined fund. Development team can take advantage of the document prepared by lessons learned team to avoid any bug during development, which would again speed up the project development. So target would reach in time and also company would get this is the utility where lessons learned works best.

### 4.20.5 Cross Utilization of Lessons Learned




**PROJECT -- 1** **PROJECT – 2**

**FIGURE-27 CROSS UTILIZATION OF LESSON LEARNED**

Lessons learned of project 1 is also utilized in project 2 also even they are of different type and not running at the same time. Lessons learned documents can be used any time in any project in future for reference.

This is also best utilization of lessons learned concept, as the documents prepared in any project can be utilized any time in future project.

### 4.20.6 Lessons Learned Issue regarding Phases of SDLC

There are lots of issues raised during the development of the project. All those issue which are important to be remembered for long term to be benefited to other forth coming projects, should included in lessons learned document. Some of the issues are described below.

#### *4.20.6.1 Projects initiation and planning phase*

This is the first phase of the project development. In this phase identify the need for a new or improved system. Information requirements are identified and project has to meet this requirements. Customer defines need for the project and these needs are prioritized and translated into plan to develop the project.

Here, problems are identified and investigated further and development planning is design according to them. The project leader and team will produce specific plan to develop the proposed project, which the team will follow using the remaining SDLC phases.

Some issues need to include in lessons learned are given below.

**1) Observation**: A new project should be developed rather than making alteration of current project.

**Discussion:** The altered system is still not working properly. Same issue of security of information is there as before. Technology is improved to make the information secure but still the system is lacking in information security. Still there are loopholes in information security with the software which can't make software reliable to use.

**Effect on organization:** As the information is not secured yet, there is no use of investing in new technology. It seems totally a waste of time and money of the organization. Still it needs to improve the project.

**Analysis:** By analyzing this issue, it is clear that it must be identified that the there must be checklist to identify whether there is need for new development of the project. Identify the factor demanding new project or continue with the current project. Like going through with the current software where there is

lacking in features, it must be identified. Whether current software should be repaired or new software should be developed having required project.

It affects the whole project management system, if such an issue introduced during the project development. Almost the project would be fail. And the reason behind is just in failing the selection of whether to alter or develop new project.

**Conclusion:** Observation or raised issue must be checked and analyzed. Here, for this issue, find where the loopholes are in current system and develop a new system which is fully secured both in terms of software and hardware.

So, in initiation phase it must have clear that alteration of current system is worth or new system should develop with extra feature for current system. Otherwise it would be unfavorable for organization as lots of money invested in the project to alter and there is also a waste of time of all the personnel. Outcome is not as required by the organization. So all the requirement for the project initiation must be check before the starting of the project.

**2) Observation:** Lots of changes during the course of the project development as project scope are not properly understood.

**Discussion:** During the course of the project development, the lots of changes are required. These changes are due to lack of understanding of the project scope. Project scope must be known properly before the starting of the project. It is the base from where to start the project as project scope knows what you are supposed to be delivering at the end to the client, and what the boundaries of the project.

**Effect on organization:** Frequent change of the requirements in bulk would be dangerous for the project development. If project scope is not defined properly then project would be fail. There is loss in monitory terms and time.

**Analysis:** There could be two reasons for arising off this issue. One reason; does not understand the basic features of the project at the project initiation level by the project management team. Another reason; owner or user could not communicate his requirements to the project management team. Owner

or user may not have knowledge regarding software or they might be not very clear about their project feature.

It would affect almost all the management plan of project development. Again they would redesign and it would a take lot of time. All the project management plans will require redesigning.

**Conclusion:** Both owner and project management team should discuss about the project scope at very early stage of the development, which will help the organization to save the time and money also. Statement of work is prepared by the project development team. Owner or user must have to go through this document and take interest in project development planning. One of the representatives from the owner side, who know which kind of project is required, has to take interest in development of the project.

**3) Observation:** Acquiring of new resources to facilitate the project, which are not required or less required.

**Discussion:** Requirement of new resources will depend on the project itself. The requirement can be of new technology or it can be of hiring skilled person or of training. Without analyzing properly of the requirement for the new resources, if it is acquire then it would affect the cost management plan and cost analysis plan of the project.

**Effect on Organization:** Yet it would affect the cost management plan and cost analysis plan of the project. It clearly shows that it also affect the cost of the project to be developed. Ultimately the project development cost would be high.

**Analysis:** As such kind of issue will affect the overall cost of the project; the money invested in acquiring the resources might be not useful at this time but can be useful in the forthcoming project. But as of now the project management team should not block the money but identify where the money should be properly invested.

Improper cost management and cost analysis can severely impact the project initiation, development and success. So, proper analysis is highly needed when the time comes to acquiring anything new for ongoing project or new project.

**Conclusion:** Acquiring new technology or hiring a skilled resource is depends on the analysis of the project need whether it is required or not. If a skilled resource is required for this project try to find out some alternatives like to give training to the personnel. These personnel will be useful in next project also. So, this kind of cost management should be done. If personnel hire then it would add extra cost to the project. If new technology is needed and if it is costly try to find out another way. If work can't be done without the new technology then care must be taken in investing them.

One more thing can be done to reduce cost of the organization to hire personnel on contractual basis or give the contract to another company having new technology or skilled person.

**4) Observation:** Schedule management plan has got problem due to problem in hours counting of personnel.

**Discussion:** Problem found at schedule management plan. There is wrong counting of man hours in the document. The error is there, and it would affect at long term even when deliverable are decided. So, a single error at basic level will affect at very high level. So, higher level plan would go wrong. This might delay the project release.

**Effect on Organization:** Organization would not affect very highly by this issue. But surely if affect to some extent. The time is wasted by re-planning the affected project management plan. It could delay the release of the project to some extent.

**Analysis:** If man hours are not calculated properly; then it would affect another plan of the project development. All the documents depend upon the schedule management plan would prepare wrongly. Like, work breakdown structure is also wrongly prepared and deliverables are wrongly calculated. If it found at early stage then re-planning of project management can be done. It would delay the project completion and project release if found at very later stage.

**Conclusion:** The entire project management plan would be carefully prepared. A single issue would lead to the delay of project development and as well as project release.

**5) Observation:** Due to lack of communication skill project manager can't explain the whole about the project to the team. Project management team doesn't understand properly about the project scope so that they can prepare project development plan properly.

**Discussion:** Project management team needs information about the project to develop the project development plan. They directly do not meet owner or user, as they are not allowed to remain present in all meetings. So, they don't know all about the project itself. They have to take help from the project manager. Now, the issue is that project manager does not have such communication skill. So, he does not make understand properly to the team about the project. Project development team can't develop project management plan properly. He also does not communicate properly with customer who is owner of the project. He can't make customer understand about what is required in project. In such scenario, project management plan would be weak and keeps on changing till the end of the project.

**Effect on organization:** It will take lots of time to develop the project management plan as team member don't understand the project features properly. It delays the project development. As the project plan is weak, continuous changes will be there throughout the development process. Again changes will delay the project. So, this issue will affect the organization in terms of finance and good will. It's waste the time of employee.

**Analysis:** Project development team members are unable to get information from project manager as he can't explain properly about the features of the project. So, team has to try for development plan. Team members work on trial and error and prepare the development plan. This takes a lot time for development of documents. This plan is continuous change because of getting real understanding of the project throughout the development process. These changes will take lots of time. So this becomes time consuming. So project can't develop at fixed dead line. It would delay the release of the project also. The success of the project will happen only then when requirements are clear to the development team. Otherwise this project will fail.

**Conclusion:** In such scenario, replace the project manager with the personnel with good communication skill. The person should one who can communicate with customer about their requirement and also explain these requirements to the team member to develop plan for the project development. If project manager is very good at management then also it would adversely affect. So, one or two person from the team member should remain present at the time of meeting or meeting should be arranged between customer and project management whole team.

So, by doing this risk of project failure would be minimized and project will also develop at right time.

### *4.20.6.2 Analysis Phase*

Analysis is the next phase. During this phase, requirements are studied and structured in accordance with their inter relationship and eliminate any redundancies. Requirements are studied rigorously so that design can be build up in the next phase and time is not wasted in changing of the requirement. It would affect the whole process of development.

The issues which can be raised during this phase are given below:

**1) Observation:** Requirements are not brief properly so they are not understandable.

**Discussion:** Basic requirement are the root material on which the project can be build up. Without requirement project can't build up. So, requirements are the main in development. If they are not properly defined then project can't reach to success. So, it is necessary to have brief understanding of requirements. If requirements are not properly defined then that project might lead to failure, as the developed project is of no use. So before implementation for project, requirements must be in understandable form.

**Effect on organization:** If requirements are not totally understood and project is developed then it surely lead to project failure. There would be loss in financial terms and waste of time is there. This also damages the good will of the project development organization.

**Analysis:** To build good and working project, proper list of requirements are required. If any mistake is there then project would be inefficient. Requirements are the key factor for the success of the project. If it gathers well then all goes well and at the end efficient project is developed. But if requirements are not understood properly then there is lots of chance of failure of the project. Requirement gathering is an art. It must be gathered carefully. Needed requirement should be briefly understood from the customer. Otherwise it would create major problems in future.

**Conclusion:** It is required for all projects to know about the requirements. All the requirements should be known in prior to start the project. Those requirements are briefly understood as on this requirements the base of the

project is positioned. To make the project successful and efficient it is mandatory for the developer to know the requirements in the beginning of the project.

**2) Observation:** Project leader does not break down the requirements at multiple levels or project leader does not have analysis power.

**Discussion:** Project leader must have capacity to analyze the requirement. If a web site is developed for e-commerce, project leader must have capacity to analyze what kind of requirements are there. The type of facility included is also asked by project leader to the owner of the website. It means how does it look, what does it do such kind of total information should be grabbed the project leader. Only an e-commerce website is required, is not the sufficient information to develop this website. There must be information included like what the environment is in which it is going to run. Which technology is required to run this website? All the information should be gathered and analyzed whether it is ok or not.

**Effect on organization:** if all the minute details are not gathered about the requirements then it would be difficult to manage the project. Because the information is raw material on which the project is developed. If requirement are not analyzed properly then customer might not satisfied with the developed project. The project might fail. Again affect organization profitability and reputation.

**Analysis:** Project development is not only based on primary requirement but to develop efficient project, detailed information should be gathered. To make very efficient project, project leader must have foresight vision or he/she can analyze the requirements. Project leader must have such experience to identify the basic needs of the project. If small details are not gathered which are prime need of the project then the project might not successful as it should be. So, project leader must have to do proper analysis of the project requirements prior the project development.

It could also happen that requirements keep on changing in bulk during the project development. That could lead to the lots of changes during in the development. Ultimately late delivery of the software would happen.

**Conclusion:** So, it is mandatory to analyze all the basic requirements. To get success in every project all and minute requirement should be clear during the project development process. Requirements needs to be taken care to develop highly efficient project.

If it requirements are changed during the development process then it could delay the project development and also release of the project would delay. It's the project leader responsibility to analyze the requirements and according to the need of the customer prioritized them.

**3) Observation:** Customer/owner is not taking interest in the project development throughout the development process.

**Discussion:** The requirements are the basic building block of the project development. Requirements must be known to the development team prior to the project development. But it is not possible for owner to explain their entire requirements prior as they are not much clear about their requirement in starting. So, at development site one of the representatives from owner should have to remain present and should take interest in the project development to some extent. So, if project development team want to ask anything or show the deliverable it would be easy. Project development could be facilitate and at the end proposed project is exactly as the customer wanted to be. Else the project can't develop smoothly and proposed project might not meet the expectation of the project.

**Effect on organization:** Due to the owner, project development organization has to suffer. Developed project might fail. So, it would be completely loss of money and waste of time.

**Analysis:** Customer or owner, who needs the proposed software, is the main source of information. The minute information related to the software needed while the software is developed. One of the members from the owner side who knows all the requirements of the project should remain present at development site. So any issue regarding information can be solved. During the development period if any changes required from the owner's side, then also it would be easy because owner is in continuous touch with the project development. So, he/she knows about the progress of the project.

Project development team can represent the development or deliverables at all phase of the development process. So at the end, the software meets the requirement of the owner and it is as they wanted to build.

But owner is not taking interest in the project development as above said then at the end of the project development life cycle, software doesn't meet the requirement of the owner and it might be of no use of the owner.

**Conclusion:** It is required for the owner or customer, who wants to develop proposed project; should remain present at the development site. He/she must have to take interest in development process. He/she can suggest the required changes during the development process if he/she remains present at the development site. If continuous presence is not there then developer does the work according to the prior defined requirements. At the end if owner or customer wants to change something like look of the front page of the website or functioning of the website, then it will take long time to change. So, it will very difficult to manage these changes at the end.

So, customer may not satisfy with the project and project might fail. So, to develop project according the requirement, customer must have to take interest in the development process and should remain present at the development site.

**4) Observation:** The development kickoff is made earlier or pushed earlier which lower the efficiency of the developer team and affect the overall project.

**Discussion:** To develop the project first it is necessary to have clear idea about requirements. Requirements gathering should be very carefully as a wring requirement would create wrong module or wrong software. So, to develop highly efficient software requirements are needed. If lacking in information regarding requirements about the software would lead to the poor quality of the software. If management push up the development team to start developing without the concrete information about the project, that would lead either to the project failure or very poor quality software.

**Effect on Organization:** This would lead to the ineffective development of the software. The software would be of low quality which will not have required features. If the continuous changes are made then also it would be time

consuming. And it would lead to delay project development and release of the software. Profit of the organization would be affected.

**Analysis:** Project development should start after the gathering all the requirements and planning. After gathering the requirements, they are analyzed. After getting proper requirements they must have prioritized according to the customers need. Like in first prototype, project based on basic and high priority requirements. Then in next iteration the other requirements are cover up. This should be the management of information gathered as requirements. But sometimes happen that management pushes the development team to start the project when they are gathering the information about the project. Without the proper information, to develop the project would be very risky and would be time consuming as changes occurs during the project development. Team members have to work for the same thing again and again, so workload of the team members would be high. This lower the efficiency of the team members and the developed software is also of low quality.

**Conclusion:** Project manager should give some time to gather information. In time during requirement gathering management should not force to start the development. Requirement should arrange properly and then project development should start. Early development of the project, without proper requirements would lead to the project failure or poor quality project.

**5) Observation:** Project manager is lacking in providing inspiration to the development team.

**Discussion:** The project manager is the in-charge of the whole team. He/she is responsible to handle the whole team, so, he/she should be supportive of the whole team. If team members need project manager support in designing management plan he/she must be there, it team members need project manager support in project development he/she must help them. This is also the responsibility of the project manager. When continuous changes are there since beginning, team members would feel tensed. So, to make them tension free should be the main priority of the project manager. Because if team

members are in tension then they can't focus their mind in project development process and result couldn't be achieved.

**Effect on Organization:** The project might lead to the poor quality as no team members are working efficiently due to constant changes since from starting. This would down the moral of the team members. That would affect the development of the project and inefficient software is developed.

**Analysis:** Project manager is the pillar of the project development team. He/she should highly support the team member especially when team members are in need. When team members need any support regarding software development then he/she used to impart their knowledge. But this only is not the responsibility of the project manager but when constant changes are there in software at that time team members feel frustrated. They need inspiration. So the project manager should be the source of inspiration for the team members. He/she must have to motivate the team members during the project development.

If project manager is lacking in providing inspiration then team members would be unable to give their best in the development of the project. Inspiration is the high need of the team member during the development of the project. If such thing happen in the starting of the project due to so much changes in the requirements then what would happen in the next coming phases. The mental condition of the team members surely affects the quality of the project.

**Conclusion:** Project manager is the leader of the project by the position. He/she should not be the leader by position but also be the leader in all aspect as his/her task is to support the team members. If project manager can't support their team members when they need then it would adversely affect the project development. When team members feel tensed about the development process, it is the duty of the manager to inspire them, encourage them to get the best result.

### *4.20.6.3 Designing phase*

In this phase the whole structure of the system is design properly. System is divided in subsystem and then how they interrelated with each other is also design in this phase. The whole structure of the system that how does it look? What does it do? Which platform is there to require running the application? How does it work? All these questions are solved and designing format is ready. Now basis on this development would start in next phase.

There are lots of issues regarding this phase those are described below.

**1) Observation:** Team members are lacking in co-operation due to personal disputes and problems.

**Discussion:** The project development is the group process and team work. Every one of the team member has to work in co-ordination and with co-operation. Co-operative work in project is necessary to develop high quality project. So, it is necessary to have co-operation among the team member of the project. If there is no co-operation among them due to disputes and personal problems then the project can't go right. The project might lead to the failure or of very low quality project. The developed project might not meet the requirements of the proposed project. So, to develop efficient project team members have to work together with harmony and co-operation.

**Effect on organization:** The organization which has disputes among their team members; can't run long. It has very short life span due to improper work done by team members. It will affect the good will of the company and creates hurdles in the development of the company.

**Analysis:** The project development is the task completed by unity of the team members. A single person cannot develop the project which has high quality. So, any organization must have united team which can co-operate with each other in developing the project. A large project needs co-operative work of team members. If they are lacking in co-operation with each other then it could be clearly predicted that the project will not get success. With such mentality in mind if team members would work then they can't concentrate on their allotted task. So, the result will be poor. Ultimately, overall project would

be poor regarding the efficiency. It might not meet the requirements described by the customer or user.

**Conclusion:** Regarding with this issue, project manager and organization management should involve in this matter and try to solve the issue or disputes among the team members. Otherwise organization would be affected badly. So, it is required to have good co-ordination and co-operation among the team members of the organization. There must be healthy relation among the team members of the organization.

**2) Observation:** Misunderstand the customer requirement to develop the design of the project. This is due to lack of customer's knowledge that what they want in the software.

**Discussion:** Misunderstand the customer's requirement is main issue in the software development. When customer doesn't know anything about the computer system at that time the chances of misunderstanding would be high. This would also increase the chances of failure. If customer is not able to express their requirements then there must be a lots of questions should be asked and also in brief to make them understand. This would help to gather proper information from the customer.

**Effect on Organization:** Rework would be there for the team members associated with the project development. Increase time and cost for the development of the project. This would the loss in profitability of the organization.

**Analysis:** It is very common issue in the software development field. Generally, customers don't have the knowledge regarding software development. Also they don't know which type of software they required. Usually this causes the software to fail. Customers are not aware of the requirement of the software so, it would be very difficult to get information about what customer want in their software. Misunderstanding like, if a customer want website for expanding their business and they just want to advertise their product on the internet. They just don't know about to specify that e-commerce facility also should be there to in their website or even they don't know about such technology. So, when functional requirement

specification does not include this then designer who develop the software design also design according to the requirement documents prepared in analysis phase.

This could lead to the increase in cost to develop the project as again this requirement could deeply analyzed and document is prepared then design is created so, it would take also extra time to complete this process. So, time is also increase to complete this project which could lead to the late development of the project. So, late release of the project would be there.

**Conclusion:** To understand the software requirement would be the key factor to get success in any software to be succeeded. Without understanding the requirements properly development should be started. If it is done so, then surely that project will not get success and it will be of low quality. If team members are unable to understand the requirements of the customer then meeting must be arranged between customer and development team. Team members should clarify their doubt regarding requirement in the meeting and then go ahead with the development.

**3) Observation:** Using incorrect or out of date information in the development of the project.

**Discussion:** To develop the good quality project, the project development process should be done in co-ordination. The team members have to work coordinately. Here, entering in the designing phase, all the documents should be prepared in the analysis phase to match the current changes made by customer. All the documents must be up to date to their current status like system requirement specification, functional requirements specification. If it is not done even by mistake then software can't work properly as desired by customer, as the outcome of the software is completely different from the desired output.

**Effect on organization:** This kind issue has direct impact on organization's good will. As to develop high quality software co-ordination is must among the team members. Otherwise software can't develop properly and caused for software failure. This would be loss of profit and loss of time also.

**Analysis:** There can be some reason behind this issue to be raised.

If personnel could not attend the important project meeting and he/she was not informed about the changes are made in the project during the meeting.

Lack of concentration of the team member during meeting session would lead to record incorrect information to be recorded.

Documents are not updated due to pressure of work so, correct and up to date information is not there in the document.

Due to above reason incorrect and up to date information cannot be used in the development process and software which is developed would not exact as wanted or described by the customer.

**Conclusion:** As above described scenarios are there to use incorrect or out of date information in the project development. So, it is the duty of the project manager to see that all the team members have up to date information regarding the developing project. All the documents must be maintained properly containing current information regarding ongoing project.

**4) Observation:** Organization management does not support constant changes due to budget limitation.

**Discussion:** There is a specified time period is allotted to analyze the requirements give by the customer to develop the project. After analyzing the requirement, designing phase is started. All the requirements specification documents must be prepared prior to the designing phase.

In designing phase, if constant changes are made by customer would delay the project development. This could also increase the development cost of the project.

**Effect on Organization:** If organization does not support, the changes made by the customer in between of the project development then customer would not be satisfied with software he/she is going to get after development.

And if organization support then profitability ratio will be down due to increased project development cost due to constant changes made by customer.

**Analysis:** All the requirements are analyzed properly by the project management team prior starting the development process. If it is not done properly then constant changes are there to be made.

Customer can have new idea for the project so; he/she can also demand to change the project in middle of the project development process.

In such scenario, ask the organization management or committee to make the changes, if they approve the changes than team members would go ahead with the changes.

**Conclusion:** Organization management should analyze the changes to be made. If it is worth to change and fit in the cost base line made for the project, then should give permission to make changes in the project.

**5) Organization:** Failure of the project development team to be with pre-decided time line due to continuous changes made in between of the ongoing process of the project development.

**Discussion:** If constant changes are approved by the organization management than team members have to go with it. But it is very job to make changes in current ongoing process of development. So, it will take long to develop the project.

Constantly changing the project would lead to frustrate the team member and decrease the efficiency of the development team to do their work. It will also increase the development time for the project.

**Effect on Organization:** Organization would be able to deliver the project very late. As due to constant changes from customer's side would make it delay. It will also increase the development cost for the ongoing project.

There are also chances of poor quality software due to boredom acquired by the team members due to constant changes made in the ongoing project.

**Analysis:** If constant changes are there to made, it would be tedious job for the development to do. It would stun the efficiency of the developer so; they can't focus on their work, which might lead to the mistake in the project development.

So, constant changes are to be made with current ongoing project at any stage would be time consuming process. This could lead the project to be developed at very late or it does not meet the decided time line of the project.

**Conclusion:** By making constant changes in the ongoing project the team members lose their focus and concentration from their work. They can't give their best to the development task. So, the developed project would be of poor quality.

Constant changes would lead the project to be delayed in their development process. So, late release would be there.

### *4.20.6.4 Development and Testing phase*

Actual implementation would be there. All the design now converts into coding. And real software is developed which can run and give the output accordingly.

**1) Observation:** Team members are not doing well as they are not having skill with particular technology and tools which is used in ongoing project.

**Discussion:** If project is developed in latest technology available then it is an obvious problem with personnel working with the project; that they cannot give their best input to the project. Then the quality of the project would be very low. Yet they don't know much about the technology, they have learn first and then implement it. It would also take lots of time to develop the project. And also this project is not reliable.

**Effect on Organization:** Such scenario might lead the project to the failure. This will affect the company's goodwill. So, this could cause the biggest problem for the company.

**Analysis:** If the project is based on e-commerce, then latest technology is going to be used in the project. Team members are not aware of this technology. Since they are learning the technology from starting of the project but they don't acquire expertise with the technology so it might lead poor efficiency of the project. Here, with the e-commerce project user's information security is most important. If project is poor at security level then there is no use of developed project. And project might fail.

So, in this case rather expert personnel should hire or give the training to the current personnel in the beginning of the project. So, this lacking in knowledge of latest technology would overcome.

**Conclusion:** There can be two options to overcome this problem. One is to hire expertise personnel to the company or second is to train the personnel of the company. Choose one of the options by doing cost analysis. The option which give best result to the organization and beneficial to the organization should be chosen. There can be another option of this issue to hire any

personnel on contract basis which would be again cost effective to the company.

**2) Observation:** Development started earlier without completing design of the software would lead the project with lots of mistake or might lead to project failure.

**Discussion:** If development started before completing system designing there would be lost of problem in the development. As development requires system design document which could contain the format of whole system. So, if that format is not ready and development is started earlier then there must be continuous changes are there at the development time. That could lead the project to be developed with poor quality.

This poor quality of the project might lead to the project failure.

**Effect on Organization:** Software would be developed with continuous changes made at development time. This would lead the poor quality project, as mistakes might be there in software coding.

Too much mistakes could lead the project to be failed. Development cost would be very high so, profitability ratio would be decrease.

**Analysis:** As development is very crucial part of software development. It is actual implementation period where all the requirements are put into coding. Real software is created, which is going to be used for business or any purpose for which it was developed.

To develop software bug free and error free, raw material like whole system design must be generate prior. This system design contains how the software look, what does it do, its functioning, relation among different modules etc. If all these information are there then software can be smoothly developed. Not many changes are made during the development phase.

But if development started earlier before completing whole system designing, then it would lead to continuous changes during development. The chances of acquiring error in the software would be very high.

**Conclusion:** Development must be started after completing the entire system designing format and completing all the documents. This would lead

to smooth development of the project. If designing documents are not ready and started off the development would lead to lots of changes in the coding during development phase.

**3) Observation:** Due to spent lots of time in analysis and designing, developing is done so fast to reach the deadline which cause failure of the project.

**Discussion:** As lots of time is spent in analysis and designing phase, then it is necessary to make hurry in the rest of the phase to develop the project in a decided time line. So, the development should have to be done speedily, which could be the cause to retain the mistake within the coding. If errors are many, that project is of no use. That is failure of the software. In rushed environment developer can't concentrate on their work. They are not at peace of mind. They are under stress and make mistake in software development.

**Effect on Organization:** As development is done in too much of hurry, software must have lots of errors. So, that project may not work properly and meet the customer's requirement. Inefficiency of the software leads to the software failure.

**Analysis:** Project development process is done step by step. Particular time slot is allotted to every phase of the development. But due to any reason, if starting phase consume more time than decided time than rear phases must suffer from mistakes s they must process and completed so much speedily to meet the deadline fixed prior to complete the software.

This speedy work in developing stage could make lots of errors in the project, some error could be rectified and some could not identify even in hurry. So, this could lead inefficient project development itself. Project would be fail due to lots of error in it.

So, it must check that any phase is not taking more time than allotted. If it so, then team member are informed to do the work fast and complete the work very fast.

**Conclusion:** It is necessary for the project manager to manage the software development process. If team members are taking too much of time in completing task, he/she must have warn them and make them work fast so

that task of every phase is done properly and with less error. As development is the real implementation of the software so this phase should complete with full concentration. Project manager is the responsible for complete the project within decided deadline. He/she is also responsible for complete this project smoothly and with ease. Project manager should take care of completing all the phases in allotted time duration.

**4) Observation:** Some requirements are changed during the development phase, which could delay the development of the project.

**Discussion:** If requirements are changed at this stage of the development than it could be very difficult to manage with the changes in requirements. Again analysis has to be done and newly arrived requirement should have to be fitted with the current designing format. Relation with the other module of the software must have to decide. After, they are to be implemented into coding. So, such process takes lots of time to complete. This could delay the project development.

**Effect on Organization:** Changes made at this level could lead to late development of the project. Late development could delay the release also. As release is delayed then it must affect the goodwill of the organization.

Changes at development stage could lead to the have mistakes in the software. Much mistake could lead to the failure of the project. Development cost will also increase due to changes made at later stage.

**Analysis:** It is very difficult to manage changes at any stage of the development process. If changes in requirements arrive at development phase than it is very difficult to incorporate this changes in current development. Changes could be any type of like functional changes, structural changes, technological changes etc.

So, newly arrived changes must have to analyze first and then it must be fitted in designing format. After that they could be incorporated in development phase. This is a very long and time consuming process. If such requirements are there in continuous form then it would be difficult for management team to cope up with these changes. So, this could lead to delay in development of the software and ultimately delay in release also.

**Conclusion:** Only those changes are allowed to make which could make the software efficient but it should be also cost effective. If required changes are not cost effective then that changes should not incorporated in software development, otherwise this could lead to delay release of the project. Project manager must have to take care at information gathering time to collect all the information at starting phase. So, this kind of scenario is not there to deal with.

**5) Observation:** Quality baseline decided at planning stage does not meet by the developed project.

**Discussion:** The purpose of the baseline is to provide a basis for ensuring that quality can be measured to determine if acceptable quality levels have been achieved. It is important for all projects to clearly define and communicate quality standards and the quality baseline serves this purpose. Every project must have to meet the quality standard decided by the organization.

If it does not meet the quality baseline decided at planning stage then it is said to be poor quality project. This poor quality project is inefficient in working so it could lead to the failure.

**Effect on Organization:** Quality of the software is must to have with the software. If software is lacking in important quality then it would be very difficult for customer to use it. It might be fail. So, it would be completely wasting of time and money.

**Analysis:** Everyone demands quality. With respect to software terminology, quality means project has to meet the quality standard decide by the organization for the project. If project reaches to that quality then it must be said a good quality project. To get success in market project has to have quality standard set by the organization.

If it does not meet the quality standard that project would fail in market as customer can't accept it. So the for example if an organization is developing a website. It has set a standard that its website can run on any environment. So when website is developed it must have to meet this standard. Such kind of quality baseline is decided by the organization which project has to meet.

**Conclusion:** To get success in software market and to get broader market software must have to good with quality. Poor quality software does not get broad market. It might get fail. Project must be tested by the tester whether the software meets the quality standard or not. If not then it must have to recheck and find where the loophole is that does not meet the quality line. Reconstruct it and again check it till the software quality does not improve.

### *4.20.6.5 Release and Maintenance phase*

**1) Observation:** Release could be delay due to difference of opinion in marketing strategy.

**Discussion:** Marketing is one of the main key characteristic to make anything popular. Marketing strategy should be like such as it can capture the focus of maximum people. There could be multiple of marketing strategy. Some of the stake holders prefer traditional marketing like print media, conference, seminar, etc. and some of them demand for modern marketing like online marketing, telemarketing. So, it would be decided prior to which strategy is going to use among such marketing strategy. Release phase is not the right time to decide which strategy to use for marketing product as this dispute could delay the release of the product.

**Effect on Organization:** Delayed release could affect the organization reputation badly. This delayed release is due to internal disputes so, when disputes resolves at that time release of the software would happen.

**Analysis:** All the post production marketing strategy should be decided earlier. If it is decided at release time then it could delay the release of the software. Marketing makes the product known to the world. Marketing should be like this that it could catch the attention of the people. So, whichever the marketing strategy would be but it must have to be affective so, that selling market could be broad.

When such kind of issue arises, then try to find out the prior project marketing strategy. If that marketing strategy catch the broad selling market then it would be effective. Try to use it in the current ongoing project to make user aware of your software.

Generally in today's world advertisement on television and online strategy would be the best for software market.

**Conclusion:** Marketing strategy should be decided earlier when the project planning is there ongoing. Choose the marketing strategy on basis of your experience and prior project marketing strategy to get the maximum market for your current software. If the software is good but marketing is not proper

then in current world that software would not survive means not accepted. Today's world is marketing world.

**2) Observation:** Generally, organization does not pay attention to the customer's problem after some time of software handed over to the customer.

**Discussion:** It happens most of the time that after passing so much time after handed over the software to the customer, development team would be busy with another project. Though maintenance team would be there to maintain the project but sometime maintenance team also need some help from development team in some critical issue. So, organization would have to pay attention to this and try to solve the customer's problem.

**Effect on Organization:** Satisfaction of customer should be the main motive of any organization. If customers are not satisfied then it would be difficult for any organization to survive in this competitive world. It could damage company's goodwill.

**Analysis:** Company's goodwill is the main criteria for any business. Goodwill should be of such kind that product would be sold by the name of the company. Every stakeholder of every project should keen with their interest to maintain company's goodwill. So, when customer has problem regarding the software then organization must have to pay attention to them. It should not behave like nothing concerning about that. Organization must make customers comfortable to come with their complaints. All the customer should be appreciated for complains and company should also look after their complaints. After if company is survive then only new project would be developed.

**Conclusion:** All the involve stakeholder must have to pay attention to the customers problem and try to solve them as early as possible. As for any organization customer is everything because without customer's need no organization would survive. So, to maintain customer satisfaction would be the prime focus of any organization.

**3) Observation:** Review should be gathered from the potential user of the software.

**Discussion:** When any software release or handed over to the customer, review is coming from every user. All the users are interested in giving their review. That review could be genuine or cannot be genuine. So, it would not be done to focus on each review and change the software according them. But all the review is collected and then checked. Only potential review needs to be checked and needs to be work upon them. Generally potential review can be gathered from potential users only.

**Effect on Organization:** Looking after the potential review would make the perfect changes in the software. This will be beneficial for software to make it very efficient. This would be beneficial for user to have efficient software and it would be beneficial for organization to increase their market in software which could again increase the goodwill of the company.

**Analysis:** Generally, potential review could be gathered from the potential user. As review is collected and checked, it must have to implement accordingly. As potential user has wide experience of the software and they could be known about the requirement of the general user and how batter they could be implement further. If such kind of changes is required then it could be done with software. That might bring good and very efficient software to use.

Review could be common also. That review should be eliminated from the list to implement.

**Conclusion:** Review from the user can be the best judgment for the software. If software is working properly or not, that only user can give review. The user can suggest for the required changes. If that changes are genuine and need to look after them then it could need to focus and changes would be made there. This might make software more efficient than before.

So, potential review must need to have attention to make changes in the software.

**4) Observation:** Review should be analyzed properly before doing any changes in software.

**Discussion:** Once software is released or handed over to the customer, customer would give reviews regarding the software. Reviews can be good or

it can be bad. It could be potential or weak. All the review should be checked as if they are potential and worth to have changes in the software accordingly then changes should be made with the software. Without analyzed properly, users review should not be implemented properly. May be they have wrong impact on software. So, care must be taken that before implementing the review gathered from the user, they must be analyzed properly.

**Effect on Organization:** Review is the potential measure of the software. If review gathered from the user is strong enough to make the software efficient than it could be implemented. This would make the software with very high quality. So, organization would get benefit from the software selling market as well as the goodwill also increase. So, profitability ratio also increases.

**Analysis:** After release of the software, generally every user gives their review to the organization which has developed the software. This review could have potential to make the software very efficient and of good quality. Some of the review does not have much impact on the software if implement it. So, analysis should be made after getting review from the user and needs to analyze that how much efficient impact it has on the software if it is implemented.

If the software quality is improve then that review must be implemented on the software. It is worth to invest time and money to make changes to the software. If it doesn't impact so much then changes should not be made with the software. Then it would be completely a waste of money and time.

So, to get benefit from the review by implementing them they must have to analyze prior before implementing them. As not all the reviews are having potential to make the software efficient.

**Conclusion:** So, review must be analyzed properly before implementing or experiment them with the software. They might be dangerous also or they can have potential also. So, it would be necessary to analyze each review before implementing them.

**5) Observation:** Software development process is lacking in post development marketing process.

**Discussion:** It is the marketing strategy which could get focus of the people to the product. So, to make the people aware of the product which is new in the market; marketing should be done properly. Post development marketing is the most important as it would be done just before sometime of release. So, people could have knowledge about upcoming software. But if there is lacking in marketing process due to unavailability of fund or whatever may be the reason is, people cannot aware of the software to be launched. So, software selling does not catch the broad market.

**Effect on Organization:** Today's world is advertising world. Marketing of the product would get the broad market for the software. If marketing process is not done properly then it might not catch the people attention towards the product itself. It would cut the profitability ratio of the organization. Sometime it would make the project fail.

**Analysis:** Marketing of the product is the first step to get attention of the people towards the product. If marketing process is made smoothly then software would get broader market for selling. It is necessary to give importance to the software marketing as development process. Marketing is as necessary as other process of development. So, marketing would be implemented strongly. If marketing is not done properly or effectively then there might be narrow selling area would be cover up by the software. It would directly affect the profit of the organization.

**Conclusion:** Marketing strategy should be strong to catch the attention of the newly developed software. If marketing is not done properly then no one in the market would be known about the software. So, to make aware about the software to sell, marketing strategy would be so strong to prepare the people to buy the software. Marketing should be given as importance as other process of the development.

# 5. Implementation and Analysis

This chapter includes actual or real implementation of the model given by the researcher and analysis of that implementation.

The purpose of this research work is to make the software development process fast and cost effective.

## 5.1 DEVELOPMENT PROCESS BEFORE STUDY

There are lots of software development process models available in the market to develop the software in current scenario. All the models have their own procedure to follow for developing the software. All have their basic criteria for getting selected.

Like, if some software project is too long to develop means it requires a long time to be developed and all the requirements are summarized prior to the starting of the software development, in such scenario waterfall and v-shaped model can be used. But if not all the requirements are sum up in the starting of the development process and might be changed during the development process, in such scenario spiral model can be selected.

So, all models have their certain criteria to be selected to develop specific project.

Now a day, agile models are very popular as these models can be used for fast development of the software. Lots of models like RAD, JAD, and XP etc are available in the market. They also have their own technique to follow.

By using such models, the software seems to be developed of good quality software at defined timeline and within defined cost. But is it possible all the time? No. There are lots of projects which were not successfully developed or yet if they developed then also they are having poor quality.

There are lots of issues or hurdles arise during the software development process. If solutions are not finding properly then it might lead to the poor

quality software. But if the root cause analysis of that particular problem is proper then solution would be ultimate.

So, during this development process of software by using any development process model, process might come across lots of issues those need to be solved.

Each issue which comes across during development process since starting phase to release phase, needs to be solved to make good quality software.

Some issues can be solved quickly as if they are simple but some issues are not as easy to solve them. They need some extra time to solve them, which could lead to delay the software development as rework has to done by implementing the ultimate solution.

Implementing new solution would take some time. So, it might delay the release of the software.

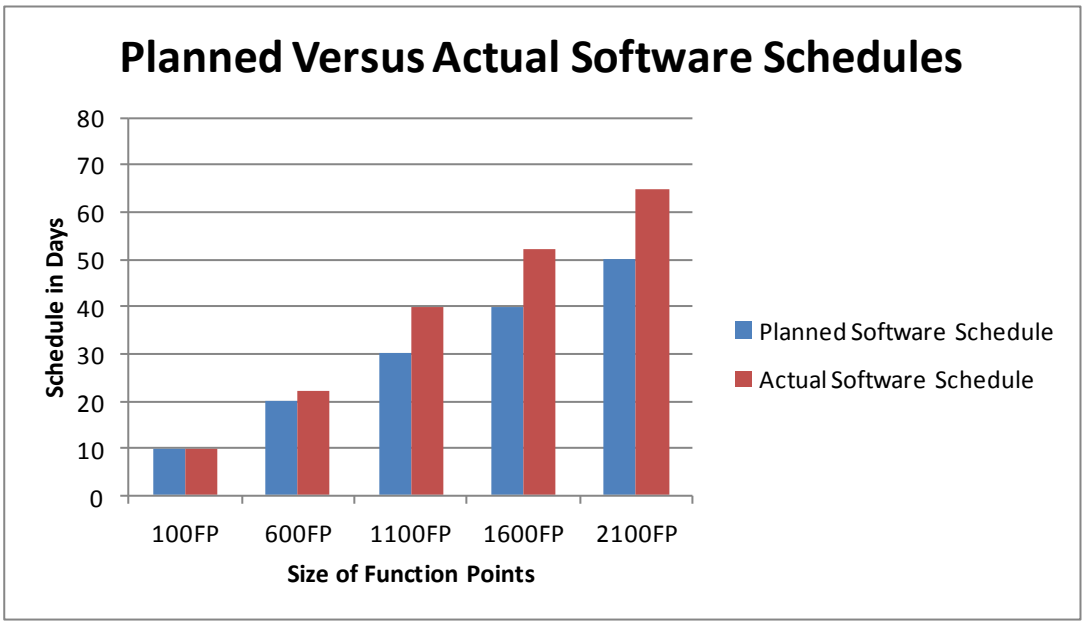



**FIGURE – 28 PLANNED VERSUS ACTUAL SCHEDULE**

In above, chart tentative data is used, as cited in a survey[46]. That shows in starting of the project development the project schedule works as planned. But as development goes on there is delay to meet required deadline. So, it

[46] http://nas.uhcl.edu/Boetticher/SWEN5430/WhySoftwareProjectsFail_CapersJones.pdf

does not work as planned. So, another bar shows “Actual Software Schedule” that is in red color while blue color is for “Planned Software Schedule”.

## 5.2 SUGGESTIONS AS A RESULT OF RESEARCH WORK

There could be lots of reason which would delay the project. Reason could be any social or technical like inaccurate estimating and schedule planning, without getting all requirements, not understanding requirements, poor knowledge of technology, etc.

Following are the suggestions which should be adopted by the each institute which are involve with the software development industry.

1) Lessons learned should be adopted as continuous process throughout the whole software development life cycle process.

2) During this process all the successfully implemented factors which makes the project successful in terms of cost and time, and all the issues which were solved should be identified.

3) In the starting of the each phase meeting / meetings (according to schedule) should be arranged which would have discussion on the tentative issues which might come across during the upcoming phase from the past lessons learned issue. This meeting should contain the discussion in detail of each issue like analysis of that issue, why it happened if possible, which are the possible ways to overcome that issue and choose the best one. It is kind of warning for the issue, which might arise during the ongoing phase.

4) At the end of the each phase meeting / meetings (according to schedule) should be arranged which would have discussion about what went good and what went wrong during the phase. Those good and bad issues should be discussed in detail.

5) If lessons learned team find that good and bad issues should be included as lessons learned then, it should be included in lessons learned database.

6) Lessons learned should be included in software development life cycle process as a complete phase also.

7) Lessons learned should be a knowledge sharing concept. It should add value to the current software development process.

8) Lessons learned database should be well administered by lessons learn team.

9) Lessons learned database should be easily accessible to the team members who require gaining knowledge from the whole institute.

10) Lessons learned team should work with the project management team and development team.

11) Strong leadership that encourages and rewards openness and encourages a culture aimed at performance improvement as opposed to placing blame toward any employee or team member for conducting or passing through any issue.

12) Lessons learned team should periodically review and improve the lessons learned process.

### 5.3 REASONS BEHIND ABOVE SUGGESTIONS

After going through till this page one must have question about why to maintain lessons learned database as information is available on internet. Then according to the research following are the reason.

1) Generally, information we find on internet is authenticated or not. It is a big question. It can be authenticated or it can't.

2) If we accept it that the information we find on internet is the perfect solution for the particular issue, then also it is very time consuming to find out the proper information that exactly required to solve any problem by any employee. It would be tedious job if it last long for many time. That solution should be implemented then after to continue the project, which might disturb the schedule.

3) If the same problem arises after some time with any other employee than again same procedure would be implemented to find the solution.

4) Again, that would be time consuming, which could affect software development which could delay software release.

5) Time required to solve particular query (which already happen earlier to other employee with same institute) affect software release as well as cost and quality of the project.

6) Means, the matter is small. Everyone is aware. But no one cares. Lessons learned are used for side-placing of obstacles and to put the warning and milestones through the dynamism.

7) But if “Lessons learned” concept is properly implemented by such institute then it would save the employee’s time, which could be implemented in some creative work and that would be beneficial to the organization in terms of early or on date completion of the project.

8) Employee might get some new and innovative ideas for the development.

9) Ultimately, it saves employee’s time, cost effective and reliable software with good quality.

## 5.4 BENEFITS ON THE IMPLEMENTATION OF RESEARCH CONCEPT

Researcher said that Lessons Learned concept is much useful in the process of software development. This concept is implemented in a company named MetrixOne System Pvt. Ltd. The result is successful development of the software that could lead to the time and cost effective development.

Below is the comparative study of the two softwares. Both were developed at MetrixOne System Pvt. Ltd. Both the software was made in Java. And Oracle is the database to store the data.

To compare the software developed without Lessons Learned concept and with Lessons Learned concept, its estimated cost and time is considered. For that COCOMO model is used.

The COnstructive COst MOdel (COCOMO) is the model used for estimating software cost and effort. These methods use a basic formula with parameters that are determined via a historical database and current project characteristics.

The COCOMO Model is usually used and for the cost / effort estimation models. This model as a size-driven model is highly dependent upon the manager's ability to estimate the size of the software system at an early stage. **This model is generally used in union with the size estimation models such as function points.**

By using Cocomo model, effort and time estimation had decided in the project. To decide effort and time estimation following calculation is required.

**IREPORT SOFTWARE:**

This is the software developed without lessons learned concept. This is having following modules.

| Modules | NO |
|---|---|
| Input | 5 |
| Output | 5 |
| Inquiry | 4 |
| Files | 6 |
| Interface | 5 |

**TABLE – 9 MODULES OF IREPORT SOFTWARE**

Those five items decides the ultimate complexity of the application. Take the weighted sum of these counts to get the number of function points (FP) in a system. Following table show weight of the particular characteristic[47].

[47] Software Engineering- A practitioner's Approach, Roger S. Pressman. McGrawHill Publication.

| Characteristic | Weight |
|---|---|
| No of Inputs | 4 |
| No of Output | 5 |
| No of inquiry | 4 |
| No of Files | 10 |
| No of Interface | 7 |

**TABLE – 10 WEIGHTS AND CHARACTERISTIC**

Suppose Java needs 30 LOC per FP. The software with above modules has FP equal to:

**FP = ( 4*5 ) + ( 5 * 5 ) + (4 * 4 ) + ( 6 * 10 ) + ( 2 * 7 ) = 135**

And estimated size would be equal to:

**Size = FP * 30 = 135 * 30 = 4050, which is approx. 4000 DSI[48].**

Here our software is small so, researcher has used organic mode for estimation.

The Basic COCOMO Effort and schedule equations for organic mode software projects are:

***SM[49] = 2.4 * ( KDSI )$^{1.05}$***

***TDEV[50]= 2.50 * ( SM )$^{0.38}$***

So, for above IREPORT SOFTWARE

Effort and Productivity would be

Effort:

**SM = 2.4 * (4)$^{1.05}$ = 10 Staff-Months**

Productivity:

**4000 KDSI / 10 SM= 400 DSI / SM**

[48] DSI means Delivered Source Instruction
[49] SM means Staff Month
[50] TDEV means Time for development

Duration and Staffing is below

Duration:

**TDEV = 2.50*(10)$^{0.38}$ = 5 Months**

Staffing:

**10 Staff-Month/ 5 Months = 2 FSP (Full time equivalent Software Personnel)**

So, cost estimation could be done by what amount we pay to our employee.

| Effort | 10 Staff-Months |
|---|---|
| Productivity | 400 DSI / SM |
| Duration | 5 Months |
| Staffing | 2 FSP |

**TABLE-11 RESULT OF COCOMO MODEL FOR IREPORT SOFTWARE**

Above result is the estimation done by the project manager at very initial stage of the development.

IREPORT SOFTWARE was started at 08/06/2012 and its target was to complete 25/10/2012. The project was to complete within 5 months (100 working days). Below is the schedule to complete the software development. Due to some activity had to run parallel the schedule is having total 88 days.

| Task no | Task Name | Days Needed | Predecessor |
|---|---|---|---|
| | Phase -1 Requirement Determination | 7 | |
| 1 | Gather Basic Requirements | 4 | - |
| 2 | User Transaction Requirements | 2 | 1 |
| 3 | User Decision Requirements | 2 | 1 |
| 4 | Whole System Requirements | 5 | 1 |
| | Phase -2 Initiation & Planning | 10 | |
| 5 | General Analysis | 8 | 4 |
| 6 | Organization Impact | 1 | 5 |
| 7 | Architectural plan/Technology Migration | 1 | 5 |
| 8 | Project scope | 2 | 5 |
| 9 | Feasibility study | 2 | 5 |
| 10 | Process planning document | 8 | 7,8,9 |
| | Phase-3 Analysis | 10 | |
| 11 | Structured Analysis | 10 | 10 |
| 12 | Dataflow diagram | 4 | 11 |
| 13 | Data dictionary | 5 | 11,12 |
| | Phase-4 System and Object Design | 13 | |
| 14 | Design of Input / Output | 5 | 13 |
| 15 | Design of File | 5 | 13 |
| 16 | Design of Database Interactions | 5 | 15 |
| 17 | Design of Controls | 4 | 14 |
| 18 | Design of system specification | 8 | 14,15,16,17 |
| 19 | Design of object | 5 | 18 |
| | Phase-5 Development & Testing | 43 | |
| 20 | Build Software | 25 | 19 |
| 21 | Test software | 15 | 20 |
| 22 | Document bugs | 5 | 21 |
| 23 | Fix found bugs | 10 | 22 |
| | Phase-6 Release & Maintenance | 5 | |
| 24 | Release software | 2 | - |
| 25 | Train the user | 1 | - |
| 26 | Test software at user side | 2 | - |

**TABLE –12 SCHEDULE FOR IREPORT SOFTWARE**

There are some issues raised during the development process. Due to the problem, that took so much time to solve the software delayed. And for that reason the software could not meet its release date. It means it could not be finished at the decided time.

Below is some issues described which came across during software development.

Issue 1) Problem occurred during storing multiple data in a table even though query is right.

This issue occurred at runtime. Data should be updated in single transaction and update but this gives runtime errors.

Check for all the field write in query. If there is any problem then this kind of error occurs but after checking all the field of the query, they are ok. And after running this query again this problem is remained.

Now the size of the storing field is minimized or it may divide into multiple queries. After implementing this action data could be stored in proper format into the table. And expected result could be achieved.

To solve this issue it had taken 4 hours. It was almost half of the time of the working day.

Issue 2) Oracle hanged after running XML query in version 11.2.0.3.0

Oracle hanged by running this query. It was done so many times but result was not changed. The entire time oracle was hanged when this query run.

Oracle was again reinstalled and this query was run but no change in result. Now, find oracle patch for this bug and installed Oracle Patch P13477790. As a result, now that query would run smoothly without any bug.

To do this first back up of the database was taken then stop all the services. After this install the patch and then start all the services and run that query. Now that query worked properly.

To solve this issue, time consumes of 3 days.

Issue 3) Time was wasted during the meeting at the development stage for weekly meeting.

As, the development progress all the team members aware with their work and set with the new development software and environment. In the starting of the phase, meeting would make them easy to have communication with each other and co-operate with each other.

During meeting, our project manager would get the feedback of the status of the current project. But weekly meeting of 1 hour at later stage was wasting the time of team members since last two weeks.

So, after passing one month like this, the weekly meeting changed to fortnight meeting of 1/2 hour. So, time could be saved and team members could put their effort to complete their task.

This should be decided at planning stage that in starting phase weekly meeting would require but at after development start meeting should be once in fortnight of ½ hour.

This issue had taken 1 hour of each team member. Again this is the loss of the time.

Issue 4) File uploading could not being done.

Generally file uploading is done by passing all the fields to the next page using req.getParameter() method in java.

But this method is not working here. So, FileItemFactory class is used and array is created so, all the items which are passing, should be stored in array. Directory should be created and in that directory file should be stored then write the logic to upload the file.

To solve this issue it needed 2 days.

Issue 5) File could not be downloaded through our website

Here, setHeader() method is used which is the instance of HTTPServletResponse interface then use file handling logic to download the file.

1 day was occupied to solve this issue.

Above described issue shows that much time is wasted in finding out the solution, so for this software schedule is delayed and lead to the late software release.

"S**uch kind of issue must be recorded in the document, which could help the other developer to find the way from problem raised during software development.**"

**"This would help a lot the developer in the same iteration or another developer. Such kind of document saves the time of the developer which could be waste in finding the solution of the problem which got solution at prior. This would lead to the fast development of the software and saves time of the developer. This saved time, the developer could invest in some more creative work with the software development".**

Below is the information which shows how much time is actually taken to complete the phases.

| Phase Name | Days Needed to complete the phase | Actual time taken to complete the phase |
|---|---|---|
| Phase -1 Requirement Determination | 7 | 10 |
| Phase -2 Initiation & Planning | 10 | 12 |
| Phase-3 Analysis | 10 | 12 |
| Phase-4 System and Object Design | 13 | 15 |
| Phase-5 Development & Testing | 43 | 50 |
| Phase-6 Release & Maintenance | 5 | 5 |
| Total days | 88 | 104 |

**TABLE –13 DECIDED AND ACTUAL DAYS TO COMPLETE PHASE FOR IREPORT SOFTWARE**

The project should have to finish within 88 days. But due to time needed in solving the issue would delay the project release by 16 days. If provision is taken to take precaution in advance by knowing the tentative issue, this delay could be avoided. Following is the chart representing the above fact.

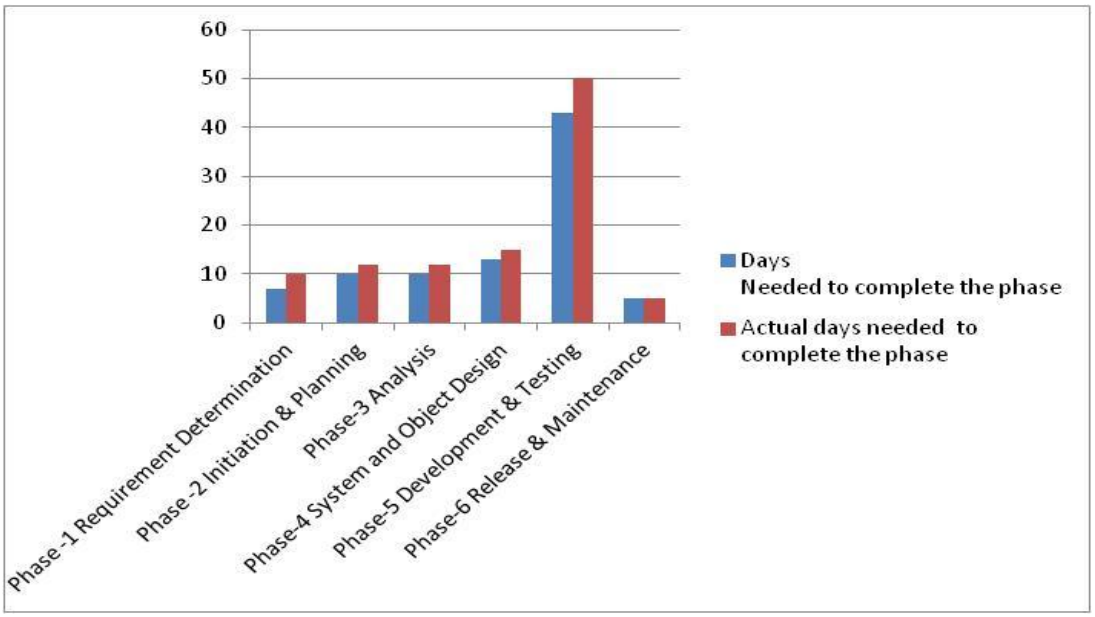



**FIGURE –30 COMPARISON OF ACTUAL DAYS AND DECIDED DAYS FOR IREPORT SOFTWARE**

Above chart clearly shows that actual days needed to complete the phase is more than decided. So, this shows that software would be developed late and would delay the software release.

**ADANI SOFTWARE:**

This is the second software. The project was started on 01/12/2012 and would be finished on 20/1/2013. This was developed using lessons learned concept. This is having following modules.

| Modules | NO |
|---|---|
| Input | 7 |
| Output | 7 |
| Inquiry | 5 |
| Files | 6 |
| Interface | 6 |

**TABLE – 14 MODULES FOR ADANI SOFTWARE**

Those five items decides the ultimate complexity of the application. Take the weighted sum of these counts to get the number of function points (FP) in a system. Following table show weight of the particular characteristic[51].

| Characteristic | Weight |
|---|---|
| No of Inputs | 4 |
| No of Output | 5 |
| No of inquiry | 4 |
| No of Files | 10 |
| No of Interface | 7 |

**TABLE – 15 WEIGHTS AND CHARACTERISTIC**

Suppose Java needs 30 LOC per FP. The software with above modules has FP equal to:

**FP = ( 4 * 7 ) + ( 5 * 7 ) + (4 * 5 ) + ( 6 * 10 ) + ( 3 * 7 ) = 135**

And estimated size would be equal to:

**Size = FP * 30 = 164 * 30 = 4920, which is approx. 5000 DSI.**

Here our software is small so, researcher has used organic mode for estimation.

[51] Software Engineering- A practitioner's Approach, Roger S. Pressman. McGrawHill Publication.

The Basic COCOMO Effort and schedule equations for organic mode software projects are:

$$SM = 2.4 * ( KDSI )^{1.05}$$

$$TDEV= 2.50 * ( SM )^{0.38}$$

So, for above IREPORT SOFTWARE

Effort and Productivity would be

Effort:

**$SM = 2.4 * (5)^{1.05}$ = 13 Staff-Months**

Productivity:

**5000 KDSI / 13 SM= 385 DSI / SM**

Duration and Staffing is below

Duration:

**$TDEV = 2.50*(13)^{0.38}$ = 6.6 Months**

Staffing:

**13 Staff-Month/ 6.6 Months = 2 FSP (Full time equivalent Software Personnel)**

So, how much is to be paid to the employee, the cost estimation could be calculated.

| | |
|---|---|
| Effort | 13 Staff-Months |
| Productivity | 385 DSI / SM |
| Duration | 6.6 Months |
| Staffing | 2 FSP |

**TABLE – 16 RESULT OF COCOMO MODEL FOR ADANI SOFTWARE**

Above result is the estimation done by the project manager at very initial stage of the development. Schedule for the ADANI SOFTWARE is given below.

| Task no | Task Name | Days Needed | Predecessor |
|---|---|---|---|
| | Phase -1 Requirement Determination | 10 | |
| 1 | Gather Basic Requirements | 5 | - |
| 2 | User Transaction Requirements | 3 | 1 |
| 3 | User Decision Requirements | 3 | 1 |
| 4 | Whole System Requirements | 5 | 1 |
| | Phase -2 Initiation & Planning | 12 | |
| 5 | General Analysis | 8 | 4 |
| 6 | Organization Impact | 1 | 5 |
| 7 | Architectural plan/Technology Migration | 1 | 5 |
| 8 | Project scope | 2 | 5 |
| 9 | Feasibility study | 2 | 5 |
| 10 | Process planning document | 7 | 7,8,9 |
| | Phase-3 Analysis | 11 | |
| 11 | Structured Analysis | 10 | 10 |
| 12 | Dataflow diagram | 5 | 11 |
| 13 | Data dictionary | 6 | 11,12 |
| | Phase-4 System and Object Design | 13 | |
| 14 | Design of Input / Output | 5 | 13 |
| 15 | Design of File | 5 | 13 |
| 16 | Design of Database Interactions | 5 | 15 |
| 17 | Design of Controls | 5 | 14 |
| 18 | Design of system specification | 8 | 14,15,16,17 |
| 19 | Design of object | 7 | 18 |
| | Phase-5 Development & Testing | 48 | |
| 20 | Build Software | 30 | - |
| 21 | Test software | 15 | 20 |
| 22 | Document bugs | 7 | 21 |
| 23 | Fix found bugs | 10 | 22 |
| | Phase-6 Release & Maintenance | 5 | |
| 24 | Release software | 2 | - |
| 25 | Train the user | 1 | - |
| 26 | Test software at user side | 2 | - |
| | Phase -7 Lessons Learned | 5 | |
| 27 | Discussion of each issue | 2 | - |
| 28 | Complete Documentation of LL | 4 | 27 |
| 29 | Sharing of LL document | 2 | 28 |

**TABLE – 17 SCHEDULE FOR ADANI SOFTWARE**

As described by COCOMO model estimation gives the result that shows 6.6 months is required to develop this software. It means around 130 days are required.

| Phase Name | Days decided to complete the phase | Actual days needed to complete the phase |
|---|---|---|
| Phase -1 Requirement Determination | 10 | 10 |
| Phase -2 Initiation & Planning | 12 | 12 |
| Phase-3 Analysis | 11 | 12 |
| Phase-4 System and Object Design | 13 | 13 |
| Phase-5 Development & Testing | 48 | 47 |
| Phase-6 Release & Maintenance | 5 | 5 |
| Phase -7 Lessons Learned | 5 | 5 |
| Total days | 104 | 104 |

**TABLE –18 DECIDED AND ACTUAL DAYS TO COMPLETE PHASE FOR ADANI SOFTWARE**

As above table shows that if tentative issues were known in advance then we could eliminate the problem before arises. This leads to smooth development of the software. As this is achieved in this software it could achieve in any software development. Following is the chart representation of above fact.

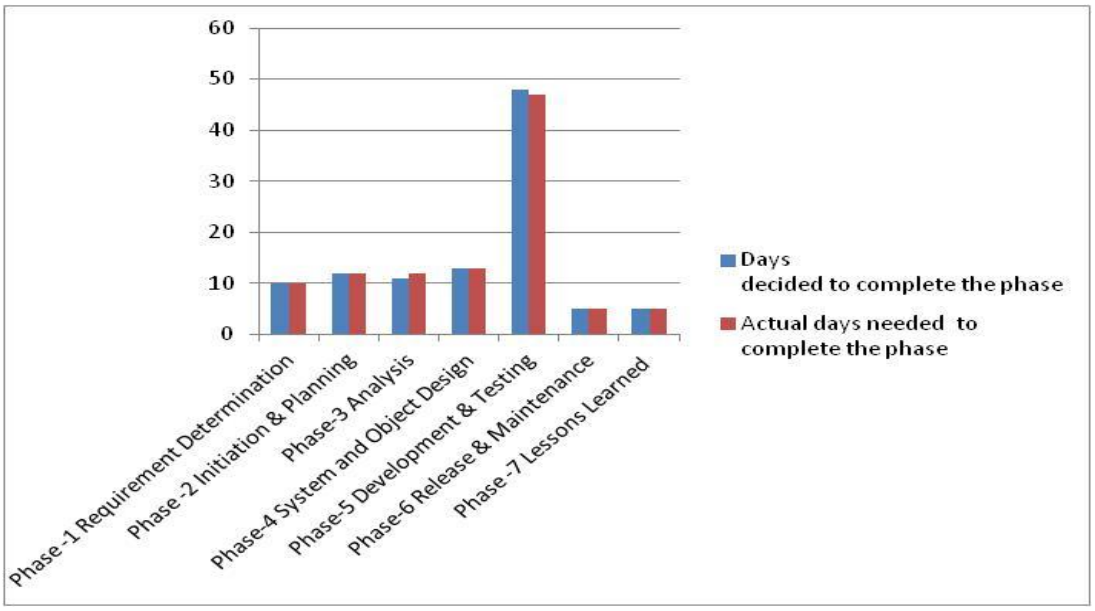



**FIGURE –32 COMPARISON OF ACTUAL AND DAYS DECIDED TO COMPLETE PHASE FOR ADANI SOFTWARE**

As above describe, two software shows that adani software was made smoothly as compared to ireport software. Development processes of both the software were same. The development process of ireport software had passed through some issues or problems which were raised during development process. Due to this issues and problems software release would be delayed. So, all the issues were identified and solved those issues then document those issues for the use of next upcoming software development.

Those documented issues were used in another software development, which is adani software. All the tentative issues were identified prior to all the phases of the software development. So, precaution was taken to avoid those issues or use the solved technique to implement it for better result.

So, from above result we can say that ireport software was not completed in decided time as company result shows while lessons learned concept were implemented and then adani software was developed. That software was developed within decided timeline.

### 5.4.1 Software development process with lessons learned helps the process to complete the development within decided timeline.

Below is the table shows the difference between actual days and decided days for both the software.

| Phase Name | Difference between actual and decided days ireport software | Difference between actual and decided days Adani software |
|---|---|---|
| Phase -1 Requirement Determination | 3 | 0 |
| Phase -2 Initiation & Planning | 2 | 0 |
| Phase-3 Analysis | 2 | 1 |
| Phase-4 System and Object Design | 2 | 0 |
| Phase-5 Development & Testing | 7 | -1 |
| Phase-6 Release & Maintenance | 0 | 0 |
| Difference between Total days | 16 | 0 |

**TABLE – 19 DIFFERENCE OF ACTUAL AND DECIDED DAYS FOR BOTH THE SOFTWARE**

Below is the chart representation of above table.

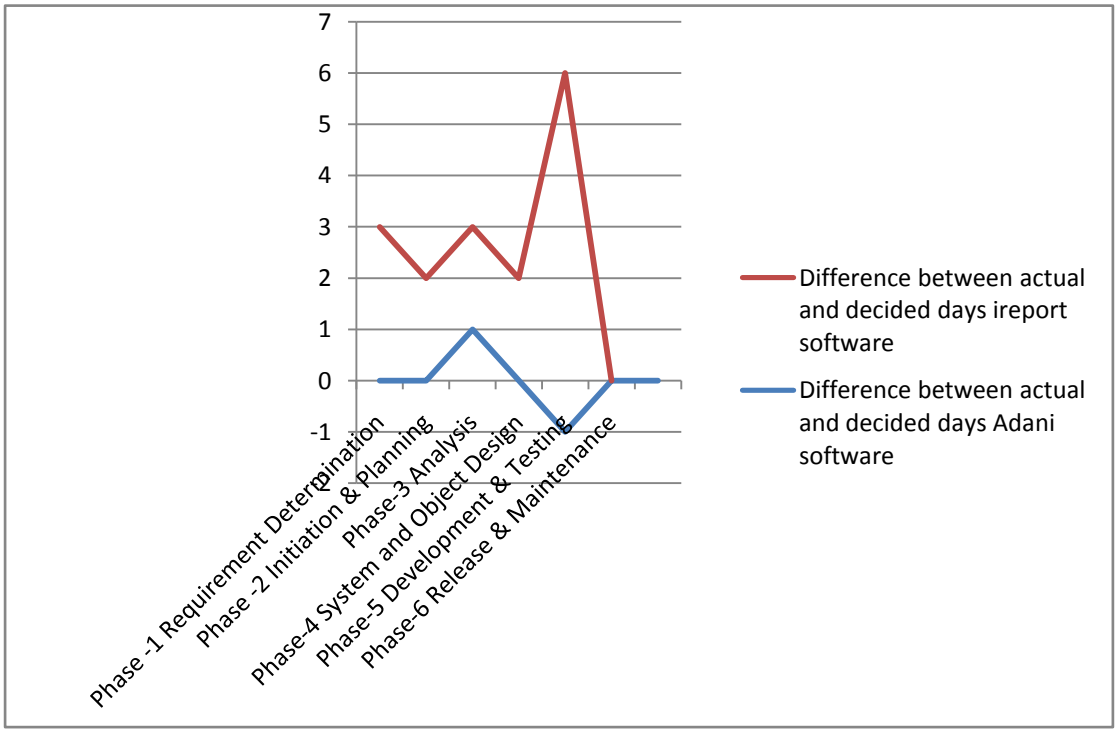



**FIGURE – 33 COMPARISON BETWEEN SOFTWARE ON THE BASIS OF THEIR PHASE COMPLETION**

Above chart clearly shows that development of adani software is smooth as the phase of the development would complete within decided time line as issues which were identified in previous software development are used as precautionary measure, while ireport software is not completed within decided time line.

So, we can see the difference between two software development processes that shows that implementing lessons learned concept to the software development process gives better result in terms of completion of software in decided timeline. That could lead to the cost effective software development and also efficient software is delivered.

### 5.4.2 Software development process with lessons learned helps the process to complete the development within estimated cost.

Ireport software is the software which developed without lessons learned concept.

Below is the table which shows the no of times the phases of the software were revised to correct the errors. There are 120 errors found during the development process.

| **Phase Name** | No of times changes are made within activities of running phase |
|---|---|
| Phase -1 Requirement Determination | 6 |
| Phase -2 Initiation & Planning | 8 |
| Phase-3 Analysis | 5 |
| Phase-4 System and Object Design | 18 |
| Phase-5 Development & Testing | 35 |
| Phase-6 Release & Maintenance | 4 |

**TABLE –20 CHANGES MADE WITHIN PHASES DURING DEVELOPMENT PROCESS FOR IREPORT**

Below is the chart which shows the no of time changes were made within the phase during the development process.

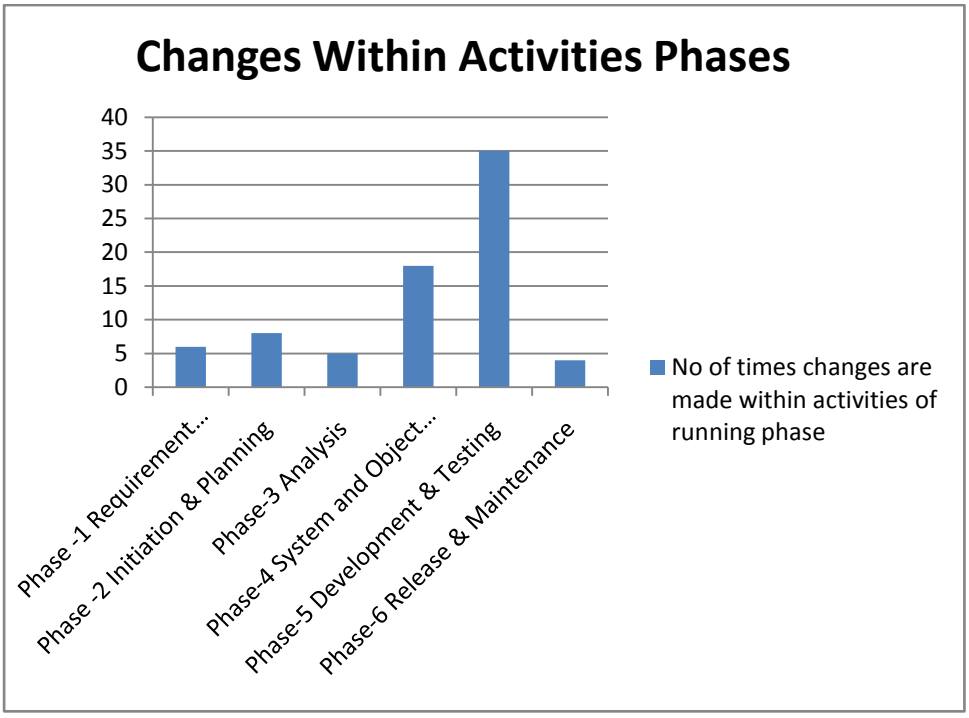



**FIGURE – 34 NO OF CHANGES WITHIN EACH PHASE DURING DEVELOPMENT OF IREPORT**

There were approximately 120 errors which were solved during the development process. To solve those errors above given no of times changes are made within activities of the particular phase as shown above.

There were 4000 lines of code in the software. To solve 120 errors it took almost 15 days. There are some errors which are serious which takes lot time to solve them. Some are moderate and some are minor errors. So, to solve these errors programmers would occupy for 15 days. So, it would have cost extra than estimated cost.

Suppose, the extra cost for 1 day would be 3000 then 15 days would cost 45000. This would be added to estimated cost of the software. So, the total cost would be high.

Adani software is the software which developed without lessons learned concept.

Below is the table which shows the no of times the phases of the software were revised to correct the errors. There are 57 errors found during the development process.

| Phase Name | No of times changes are made within activities of running phase |
|---|---|
| Phase -1 Requirement Determination | 2 |
| Phase -2 Initiation & Planning | 1 |
| Phase-3 Analysis | 1 |
| Phase-4 System and Object Design | 2 |
| Phase-5 Development & Testing | 8 |
| Phase-6 Release & Maintenance | 1 |

**TABLE –21 CHANGES MADE WITHIN PHASES DURING DEVELOPMENT PROCESS FOR ADANI SOFTWARE**

Below is the chart which shows the no of time changes were made within the phase during the development process of adani software.

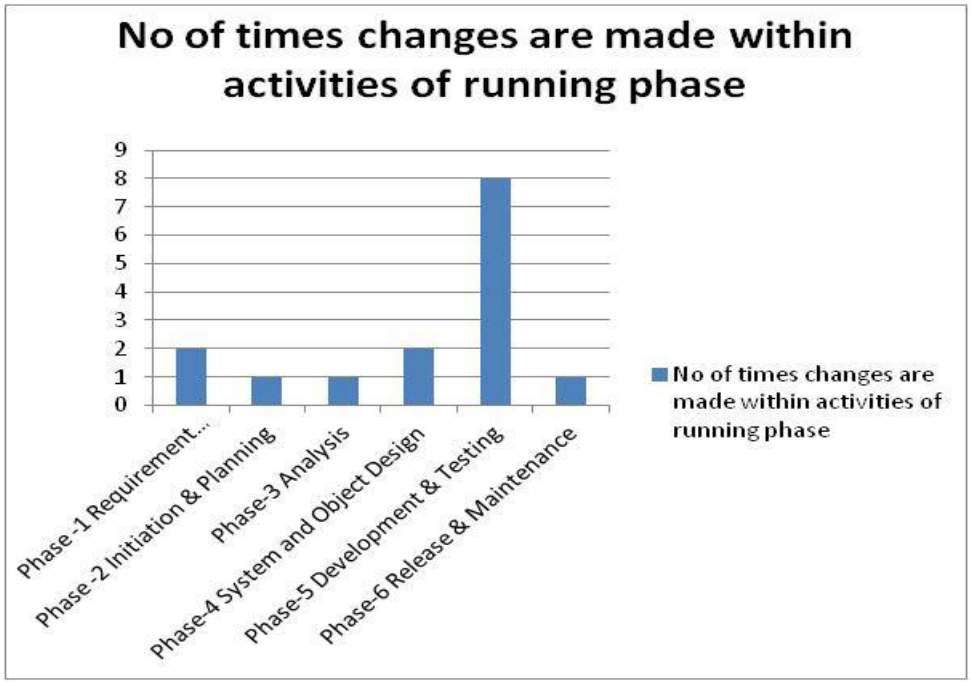



**FIGURE – 35 CHANGES WITHIN EACH PHASE DURING DEVELOPMENT OF ADANI SOFTWARE**

There were approximately 57 errors which were solved during the development process. There are 5000 lines of code with this adani software. To solve those errors above given no of times changes are made within activities of the particular phase as shown above.

There were 5000 lines of code in the software. To solve 57 errors it took almost 3 days. There are only 3 errors which are serious which takes 2 days time to solve them. Some are moderate and some are minor errors. So, to solve these errors programmers would occupy for 3 days.

Suppose, the extra cost for 1 day would be 3000 then 3 days would cost 9000. This would be added to estimated cost of the software. So, the total cost would be more compared to estimated cost.

| | **Adani software** | **Ireport software** |
|---|---|---|
| Development process | With Lessons learned concept | Without lessons learned |
| Size | 5000 | 4000 |
| Total no of errors | 57 | 120 |
| Days to solve errors | 3 | 15 |
| Serious error | 15 | 53 |
| Moderate error | 20 | 31 |
| Minor error | 22 | 36 |
| Error solving cost | 9000 | 45000 |
| Total cost | not very high | very high compared to estimated cost |

**TABLE –22 COMPARISON OF ADANI AND IREPORT SOFTWARE DEVELOPMENT PROCESS**

As above table shows that adani software required has only 15 serious errors and development would be before time so, remaining time would be utilized in error solving. Other moderate and minor errors also solved quickly as defined company's result. So, it would not cost much to solve errors.

While Ireport software development cost much than the estimation so, as there were no of errors.

So, from above table, we can say that development process that is lessons learned concept is quite better.

### 5.4.3 Software development process with lessons learned helps to develop the software having good quality.

Quality assurance is an essential activity for any business that produces the products to be used by others. Same things apply to software development industry.

Software which is being developed must have good quality like security, completeness, consistency, simplicity etc. and that should run smoothly without any bug or fault.

Below is the information about both the software regarding its quality.

As we have already seen that ireport software came across about 120 errors during software development process. Some of the errors are serious errors, some are moderate and some are minor errors. Below is the table which shows how many errors are serious, moderate and minor in a particular phase.

| Phase Name | Error In % | Error in No. | Serious errors | Moderate errors | Minor errors |
|---|---|---|---|---|---|
| Phase -1 Requirement Determination | 12 | 14 | 5 | 4 | 5 |
| Phase -2 Initiation & Planning | 15 | 18 | 10 | 5 | 3 |
| Phase-3 Analysis | 10 | 12 | 5 | 3 | 4 |
| Phase-4 System and Object Design | 20 | 24 | 10 | 6 | 8 |
| Phase-5 Development & Testing | 35 | 42 | 20 | 10 | 12 |
| Phase-6 Release & Maintenance | 8 | 10 | 3 | 3 | 4 |
| Total | 100 | 120 | 53 | 31 | 36 |

**TABLE –23 ERROR DETAILS FOR IREPORT SOFTWARE**

To decide the quality of the software MetrixOne software developer has followed statistical software quality assurance technique.

So, to decide the quality of the software, information about software errors is collected and categorized as above table.

By using information of the above Error Index of the software is counted[52].

To count Error Index below formula is used.

$EI = ( PI_1 + 2*PI_2 + \ldots. + i*PI_i) / PS$

Where, PS is the size of the software. $PI_i$ is the Phase Index. And it is calculated:

$PI_i = W_s ( S_i / E_i ) + W_m ( M_i / E_i ) + W_t ( T_i / E_i)$

Here, $W_s$ = Weighting factor for serious error = 10

$W_m$ = Weighting factor for moderate error = 3

$W_t$ = Weighting factor for minor error = 1

$S_i$ = no of serious error

$M_i$ = no of moderate error

$T_i$ = no of minor error

$E_i$ = total no of errors uncovered during the $i^{th}$ phase

First, calculation of PI for all the phases

$PI_1 = 10 (5/14) + 3(4/14) + 1(5/14) = 4.83$

$PI_2 = 10 (10/18) + 3(5/18) + 1(3/18) = 6.61$

$PI_3 = 10 (5/12) + 3(3/12) + 1(4/12) = 5.28$

$PI_4 = 10 (10/24) + 3(6/24) + 1(8/24) = 5.28$

$PI_5 = 10 (20/42) + 3(10/42) + 1(12/42) = 5.81$

$PI_6 = 10 (3/10) + 3(3/10) + 1(4/10) = 4.3$

Now,

[52] Software Engineering- A practitioner's Approach, Roger S. Pressman. McGrawHill Publication.

EI = 4.83 + (2* 6.61) + (3* 5.28) + (4*5.28) + (5* 5.81) + (6 * 4.3)/4000

= 84.06

And adani software has been passed through 57 errors.

Below is the table which shows how many errors are serious, moderate and minor in a particular phase.

| Phase Name | Error In % | Error in No. | Serious errors | Moderate errors | Minor errors |
|---|---|---|---|---|---|
| Phase -1 Requirement Determination | 20 | 11 | 2 | 4 | 5 |
| Phase -2 Initiation & Planning | 10 | 6 | 1 | 2 | 3 |
| Phase-3 Analysis | 10 | 6 | 1 | 3 | 2 |
| Phase-4 System and Object Design | 20 | 11 | 2 | 4 | 5 |
| Phase-5 Development & Testing | 30 | 17 | 7 | 5 | 5 |
| Phase-6 Release & Maintenance | 10 | 6 | 2 | 2 | 2 |
| Total | 100 | 57 | 15 | 20 | 22 |

**TABLE –24 ERROR DETAIL FOR ADANI SOFTWARE**

To count Error Index, we have to find Phase Index.

$PI_1$ = 10 (2/11) + 3(4/11) + 1(5/11) = 3.36

$PI_2$ = 10 (1/6) + 3(2/6) + 1(3/6) = 3.16

$PI_3$ = 10 (1/6) + 3(3/6) + 1(2/6) = 3.5

$PI_4$ = 10 (2/11) + 3(4/11) + 1(5/11) = 3.36

$PI_5$ = 10 (7/17) + 3(5/17) + 1(5/17) = 5.29

$PI_6$ = 10 (2/6) + 3(2/6) + 1(2/6) = 4.66

Now,

EI = 3.36 + (2* 3.16) + (3* 3.5) + (4*3.36) + (5* 5.29) + (6 * 4.66)/5000

= 60.07

Error Index is for Ireport is 84.06 and for adani software is 60.07.

So, we can conclude that adani software is having better quality than Ireport. Adani software is developed by using lessons learned concept. So, it is proved that lessons learned concept is useful to develop better quality software.

## 5.5 CONCLUSION

Major software projects have been troubling business activities for more than 50 years. Of any known business activity, software projects have the highest probability of being cancelled or delayed. Once delivered, these projects display excessive error quantities and low levels of reliability. Both technical and social issues are associated with software project failures. Among the social issues that contribute to project failures are the rejections of accurate estimates and the forcing of projects to adhere to schedules that are essentially impossible. Among the technical issues that contribute to project failures are the lack of modern estimating approaches and the failure to plan for requirements growth during development. However, it is not a law of nature that software projects will run late, be cancelled, or be unreliable after deployment. A careful program of risk analysis and risk abatement can lower the probability of a major software disaster. And this care should be taken by learning from the past experience that is covered as “Lessons Learned” in the research work.

As a proverb says “Prevention is better than Cure”, the concept of lessons learn work as preventive measure for the software development process. The outcome of this concept is successful software with complete reliability.

A Lessons Learned Process is one that is beyond the functional boundaries and allows an organization to learn from both its mistakes and its successes. An effective Lessons Learned process should prevent us from repeating our

mistakes and allow us to repeat our successes. It should be an instrumental part of any organization's overall "continuous improvement" process.

Unfortunately, very few organizations can claim they have an effective Lessons Learned process that spans their global project operations.

# 6. Conclusion

With respect to the title of the research, researcher has tried to study different software development process model, and analyzed them. After analyzing them, researcher tried to compare these entire models.

There are lots of models available to develop the software. Now days there are three main basic models available to develop the software, which researcher tried to discuss in chapter 2. Some of the points are given below.

## 6.1 UNSTRUCTURED MODEL

Development can be approached without any great attention to organising activities, resources or to the planning and analysis of what will actually be involved in delivering an item of development work.

An organisation can simply pour into a project the required elements of people, money and time then set off working in the hope that something useful will be produced. Alternatively they can enter a loop of finding and fixing issues until the product is deemed ‘good enough’.

## 6.2 HEAVYWEIGHT MODEL

Heavyweight methodologies are considered to be the traditional way of developing software. These methodologies are based on a sequential series of steps, such as requirements definition, solution building, testing and deployment. Heavyweight methodologies require defining and documenting a stable set of requirements at the beginning of a project.

## 6.3 LIGHTWEIGHT MODEL OR AGILE APPROACH

Agile as the name suggest “the quality of being agile that is readiness for motion, quickness, activity, handiness in motion”.

By this lightweight approach software development methods are attempting to offer once again an answer to the eager business community asking for lighter weight along with faster and quick software development process.

Some of the Lightweight processes are Scrum, Lean, Dynamic System Development, Rapid Application Development, Joint Application Development, and Extreme Programming.

Researcher has studied almost ten model all together of Heavyweight and Lightweight Model. All the models have their advantages and limitations. They have their specific use in the software development industry.

But in all this, researcher did not find any provision to eliminate repeatedly occurring problem during software development process. So, researcher has tried to add a concept of “Lessons Learning” with the software development process as a continuous process and also add as a phase.

Researcher has tried to develop a model which could help in smooth development of software with fewer complexities with the development. By using this model quality software can be developed with less time. As software development is very time consuming and lengthy process, so this model could be used to develop the software with ease and in limited time.

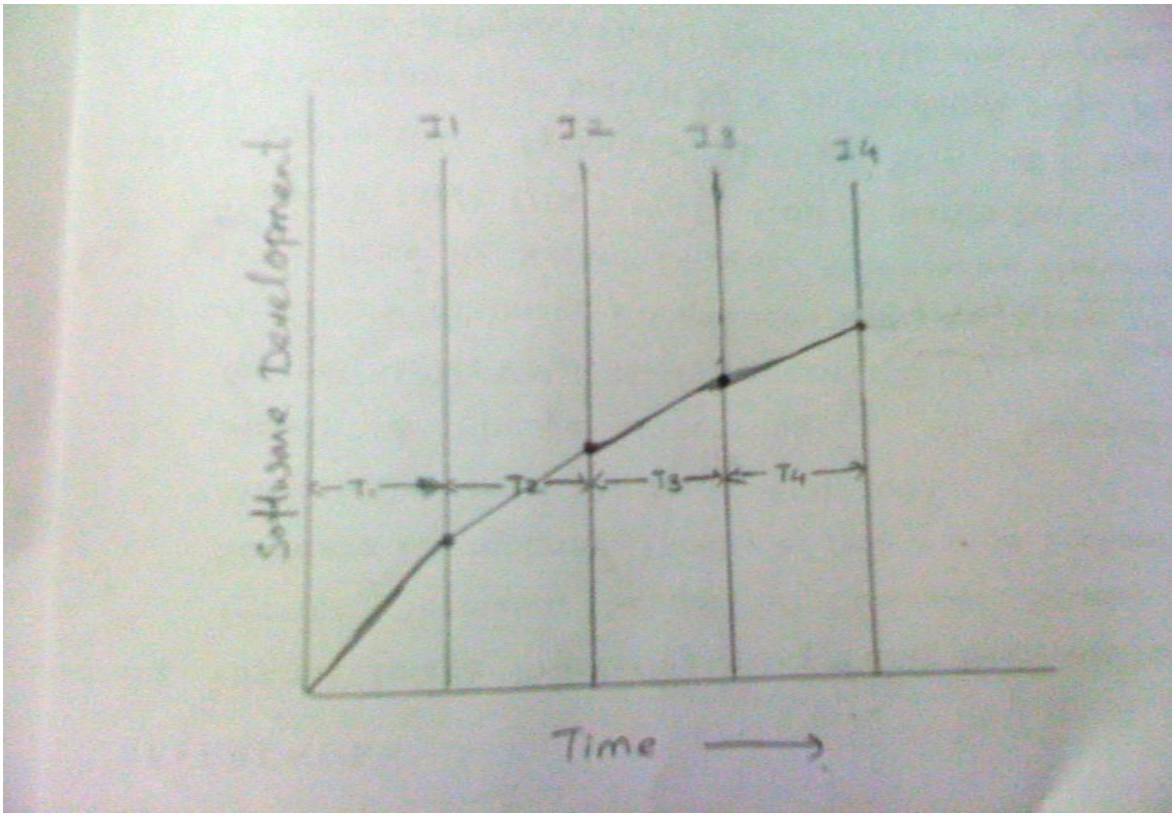



**FIGURE - 36 SOFTWARE DEVELOPMENTS GRAPH WITH NORMAL MODEL**

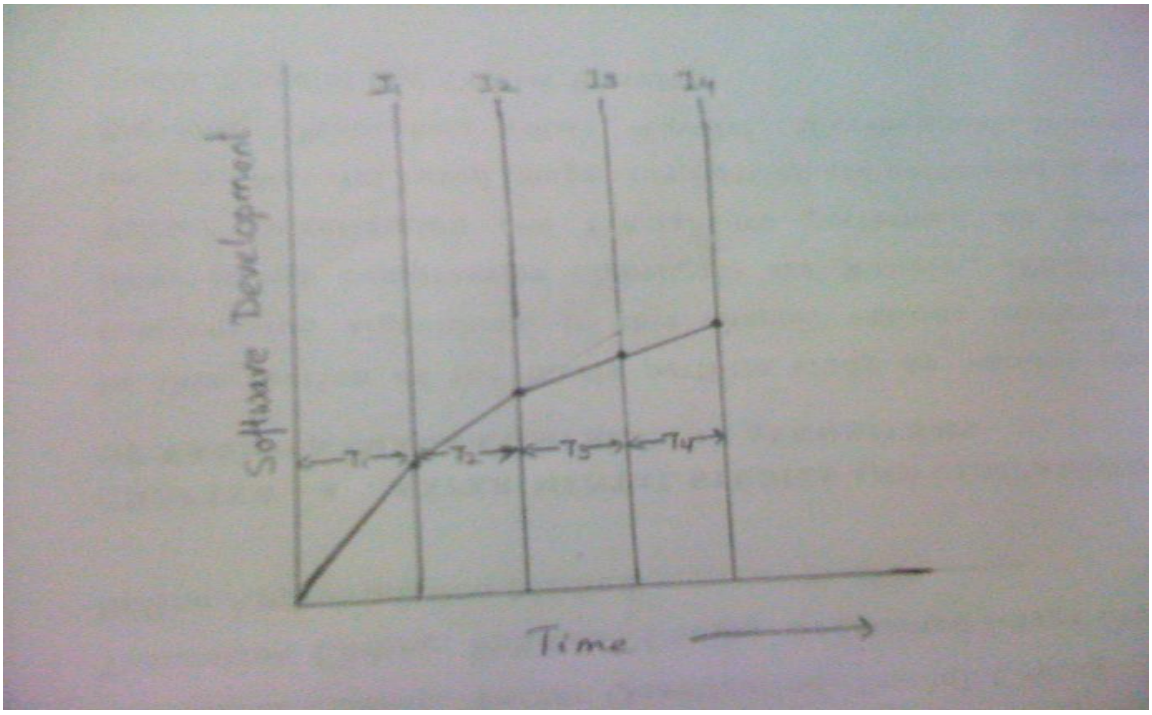



**FIGURE - 37 SOFTWARE DEVELOPMENTS WITH LL MODEL GIVEN BY RESEARCHER**

As in above given figure it is completely shown that time consumed by software development by using the model given by the researcher is low as compared to using the current software development model. This is due to learning the problems prior to the phase which could be arise during the development phase. This could minimize the development time consumed in development.

Again by using this model quality software is developed. This is the highest advantage of this research, as tentative problems are identified earlier so, elimination of that problem is done earlier. So, the development which could be achieved at later stage can be achieved in earlier stage.

In this research work, researcher has tried to develop a software development model which is useful in smooth development of the software with almost less complexity and hurdles. By using this new suggested model, developer can develop the software with ease and in less time duration. Time compresses for development as development problems are identified prior to the phase.

This research work can be useful in large software Development Company to ease the software development. By using proposed model, large software Development Company can develop quality product and support system. This quality product can be developed in very less time, with more perfection. This model helps in smooth software development process.

Next is, computer educational institute will also get benefit from this proposed model. While they are imparting training this aspect will be useful. Proper training is given so that they will educate people to produce quality software. So, they will impart knowledge to produce quality software.

## 6.4 MERITS OF THE LLMODEL

Following are the merits of the research done by researcher.

Newly developed model would help industrial people to solve issue identified during development of the product.

As Lessons learned process goes on in continuous manner with the software development, it improves the quality of the software.

Developer can smoothly develop the software. Software will be developed with fewer hurdles.

All the mistakes are recorded, so a kind of discipline is maintained not to hide any troubles through any personnel gone or any issue faced by any personnel.

It delivers quality software with highest accuracy.

It is support mechanism which supports the development process in developing problem free development.

It is continuous process to be done and data which are collected being use in ongoing project and in further iterative project.

It is interactive and live concept.

## 6.5 EXPERIENCED BENEFITS OF LLMODEL

Following are the experienced benefits of the research.

By following this model, all the tentative problems are identified in the beginning of the each software development life cycle phase.

If any issue raise during the development phase the solution would be available from the past issue.

Development time taken by the team is low as compared to following other software development model.

Quality of the software is very high in given time period.

Co-operation among team members is increased.

No one is expert. So, company will not suffer from if any of the team members left the company.

As the desired output comes in desired time so, it is beneficial in monetary teams also. Development could be finished in decided fund.

## 6.6 RECENT LIMITATIONS OF LLMODEL

Researcher has tried to develop a model which could help people who are working in industry. Yet researcher has tried her best. But there are few limitations regarding this research which is given below.

Proper verification of Data is necessary as these data would be useful in ongoing iteration or in next iteration. As data is stored in repository that would be verified properly by highly experienced person. After verification that data could be used as reference to solve any issue observed during development process.

There are limitations in number of samples as more samples are needed. As more samples are needed with respect to this proposed model to experience more accurate implementation.

If authority or responsible person neglect this concept to implement than it will lose its importance. So, all the success of this newly developed concept is dependent on well execution of this concept.

## 6.7 FUTURE SCOPE OF LLMODEL

### 6.7.1 Interdisciplinary Innovative

This concept is useful in many disciplines as this concept is useful to handle the frequent occurring error or issue. This concept is the concept to learn from past experience. As in life we are also learning from past experience. But this is our individual life. When we are working in team, each team members are having their own experience. If this experienced is shared within the team, it would be useful in the ongoing project as well as new project also. So this is the concept of sharing "**GOOD AS WELL AS BAD EXPERIENCE FACED DURING THE TIME OF DEVELOPMENT OF ANY PROJECT**".

So this concept is not limited to the software development industry but it could be very vast concept. It can be useful in any zone or in any industry where team work is there to complete the desired task.

### 6.7.2 Reference for Further Research Makers

It can be useful for further research also as all the issues are documented. If anybody wants to find another way of solution, researcher can use this document as reference for their research.

Researcher can find different ways for analysis, remedial action to be placed. Researcher uses this concept in distributed development of the project also. Again it is very vast concept.

## 6.8 BIBLIOGRAPHY

**Books**


- Software Engineering- A practitioner's Approach
  - Roger S. Pressman,- McGrawHill Publication

- Software Engineering A primer
  - Waman S. Jawadekar,- TaTa McGrawHill Publication

- Analysis and Design of Information Systems
  - James A. Senn, - McGrawHill Publication

- Object-Oriented Modeling and Design
  - James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen, - PHI publication

- Software Engineering
  - K.K. Aggarwal, Yogesh Singh, -New Age International, 2005

- **Handbook of Software Reliability Engineering**
  - Michael R. Lyu, -McGraw-Hill 1996

- **The Grand Unified Theory of Software Engineering**
  - Mathias Ekstedt, -Industrial Info Systems 2005


**Web Sites**


- http://www.veracode.com/security/software-development-lifecycle
- http://en.wikipedia.org/wiki/Systems_development_life-cycle
- http://www.sdlc.ws/what-is-sdlc/
- http://www.webopedia.com/TERM/S/SDLC.html
- http://www.techopedia.com/definition/22193/software-development-life-cycle-sdlc
- http://acronyms.thefreedictionary.com/SDLC
- http://commoninterview.com/Testing_Interview_Questions/what-is-sdlc-software-development-life-cycle/
- http://www.learn.geekinterview.com/it/sdlc/sdlc-methodology-steps.html
- http://www.benderrbt.com/Bender-SDLC.pdf

- http://www.studymode.com/essays/What-Is-Software-And-Sdlc-Need-946970.html
- http://codebetter.com/raymondlewallen/2005/07/13/software-development-life-cycle-models/
- http://istqbexamcertification.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/
- http://en.wikipedia.org/wiki/Spiral_model
- http://en.wikipedia.org/wiki/V-Model_(software_development)
- http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/
- http://www.tutorialspoint.com/sdlc/sdlc_v_model.htm
- http://en.wikipedia.org/wiki/Waterfall_model
- http://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm
- http://verification-validation.blogspot.in/2012/07/softwarelife-cycle-models.html
- http://en.wikipedia.org/wiki/Software_prototyping
- http://en.wikipedia.org/wiki/Iterative_and_incremental_development#Contrast_with_Waterfall_development
- http://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm
- http://www.selectbs.com/analysis-and-design/what-is-rapid-application-development
- http://en.wikipedia.org/wiki/Rapid_application_development
- http://en.wikipedia.org/wiki/Joint_application_design
- http://www.carolla.com/wp-jad.htm
- http://www.mountaingoatsoftware.com/topics/scrum
- http://scrummethodology.com/
- http://xprogramming.com/book/whatisxp/
- http://www.extremeprogramming.org/
- http://pmtips.net/lessons-learned-template/
- http://www.projectsmart.co.uk/lessons-learned.html
- http://www.tutorialspoint.com/management_concepts/project_lessons_learned.htm

- http://www.projectmanagementdocs.com/project-closing-templates/lessons-learned.html
- http://www.cmu.edu/computing/ppmo/project-management/life-cycle/closing/lessons/index.html

## 6.9 RESEARCH PAPER PUBLISHED IN INTERNATIONAL RESEARCH JOURNAL NAMED IJCAIT IN VOLUME – 2 NO – 1 (2013)

# INDEX

## E



## F



## G



## H



## I



## J



## L



## M

## O



## P



## Q



## R



## S

## T



## U



## V



## W



## X

| Task no | Task Name | Days Needed | Predecessor |
|---|---|---|---|
| | Phase -1 Requirement Determination | 7 | |
| 1 | Gather Basic Requirements | 4 | - |
| 2 | User Transaction Requirements | 2 | 1 |
| 3 | User Decision Requirements | 2 | 1 |
| 4 | Whole System Requirements | 5 | 1 |
| | Phase -2 Initiation & Planning | 10 | |
| 5 | General Analysis | 8 | 4 |
| 6 | Organization Impact | 1 | 5 |
| 7 | Architectural plan/Technology Migration | 1 | 5 |
| 8 | Project scope | 2 | 5 |
| 9 | Feasibility study | 2 | 5 |
| 10 | Process planning document | 8 | 7,8,9 |
| | Phase-3 Analysis | 10 | |
| 11 | Structured Analysis | 10 | 10 |
| 12 | Dataflow diagram | 4 | 11 |
| 13 | Data dictionary | 5 | 11,12 |
| | Phase-4 System and Object Design | 13 | |
| 14 | Design of Input / Output | 5 | 13 |
| 15 | Design of File | 5 | 13 |
| 16 | Design of Database Interactions | 5 | 15 |
| 17 | Design of Controls | 4 | 14 |
| 18 | Design of system specification | 8 | 14,15,16,17 |
| 19 | Design of object | 5 | 18 |
| | Phase-5 Development & Testing | 43 | |
| 20 | Build Software | 25 | - |
| 21 | Test software | 15 | 20 |
| 22 | Document bugs | 5 | 21 |
| 23 | Fix found bugs | 10 | 22 |
| | Phase-6 Release & Maintenance | 5 | |
| 24 | Release software | 2 | - |
| 25 | Train the user | 1 | - |
| 26 | Test software at user side | 2 | - |

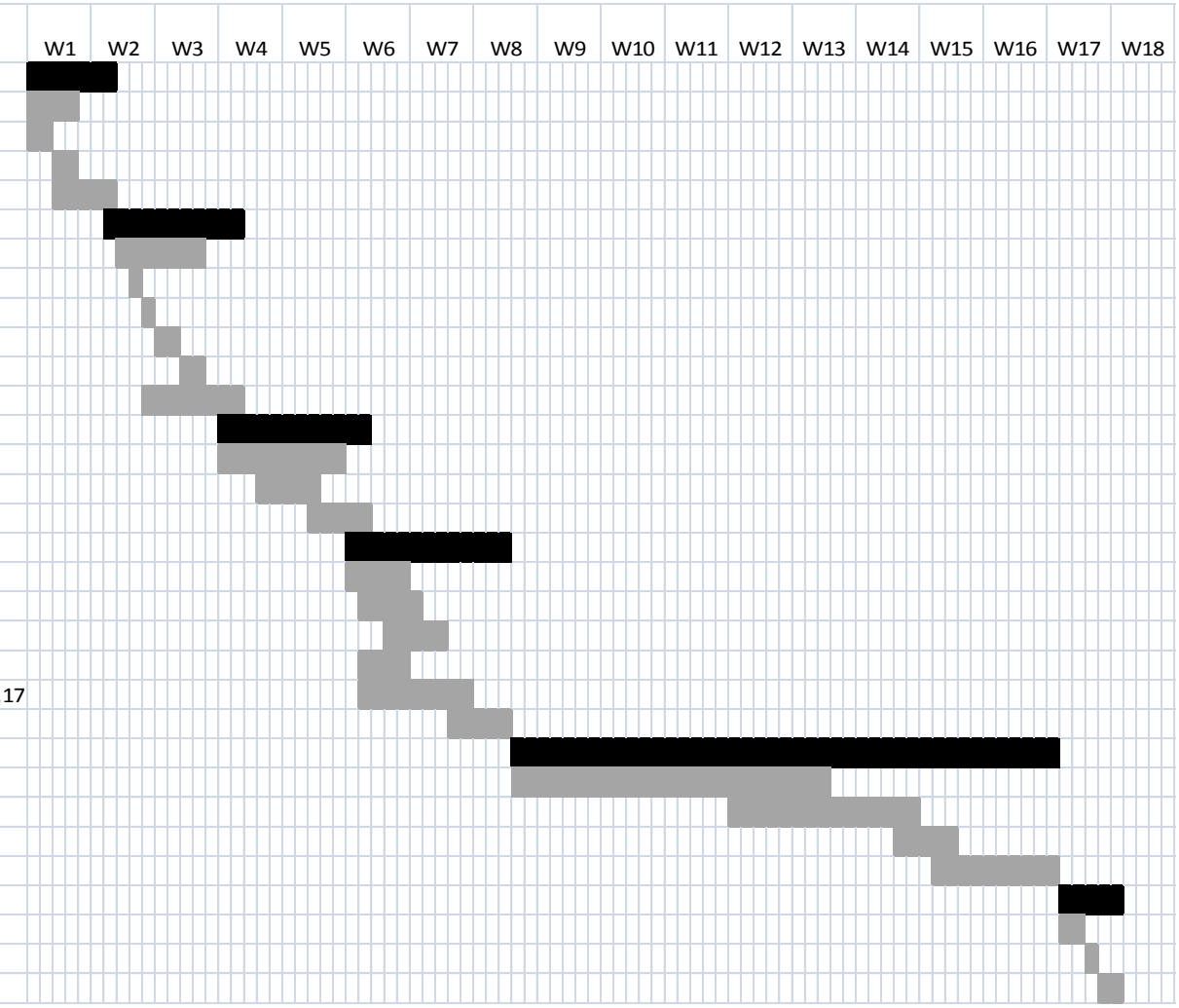



**FIGURE –29 GANTT CHART FOR IREPORT SOFTWARE**

| Task no | Task Name | Days Needed | Predecessor |
|---|---|---|---|
| | Phase -1 Requirement Determination | 10 | |
| 1 | Gather Basic Requirements | 5 | - |
| 2 | User Transaction Requirements | 3 | 1 |
| 3 | User Decision Requirements | 3 | 1 |
| 4 | Whole System Requirements | 5 | 1 |
| | Phase -2 Initiation & Planning | 12 | |
| 5 | General Analysis | 8 | 4 |
| 6 | Organization Impact | 1 | 5 |
| 7 | Architectural plan/Technology Migration | 1 | 5 |
| 8 | Project scope | 2 | 5 |
| 9 | Feasibility study | 2 | 5 |
| 10 | Process planning document | 7 | 7,8,9 |
| | Phase-3 Analysis | 11 | |
| 11 | Structured Analysis | 10 | 10 |
| 12 | Dataflow diagram | 5 | 11 |
| 13 | Data dictionary | 6 | 11,12 |
| | Phase-4 System and Object Design | 13 | |
| 14 | Design of Input / Output | 5 | 13 |
| 15 | Design of File | 5 | 13 |
| 16 | Design of Database Interactions | 5 | 15 |
| 17 | Design of Controls | 5 | 14 |
| 18 | Design of system specification | 8 | 14,15,16,17 |
| 19 | Design of object | 7 | 18 |
| | Phase-5 Development & Testing | 48 | |
| 20 | Build Software | 30 | - |
| 21 | Test software | 15 | 20 |
| 22 | Document bugs | 7 | 21 |
| 23 | Fix found bugs | 10 | 22 |
| | Phase-6 Release & Maintenance | 5 | |
| 24 | Release software | 2 | - |
| 25 | Train the user | 1 | - |
| 26 | Test software at user side | 2 | - |
| | Phase -7 Lessons Learned | 5 | |
| 27 | Discussion of each issue | 2 | - |
| 28 | Complete Documentation of LL | 4 | 27 |
| 29 | Sharing of LL document | 2 | 28 |

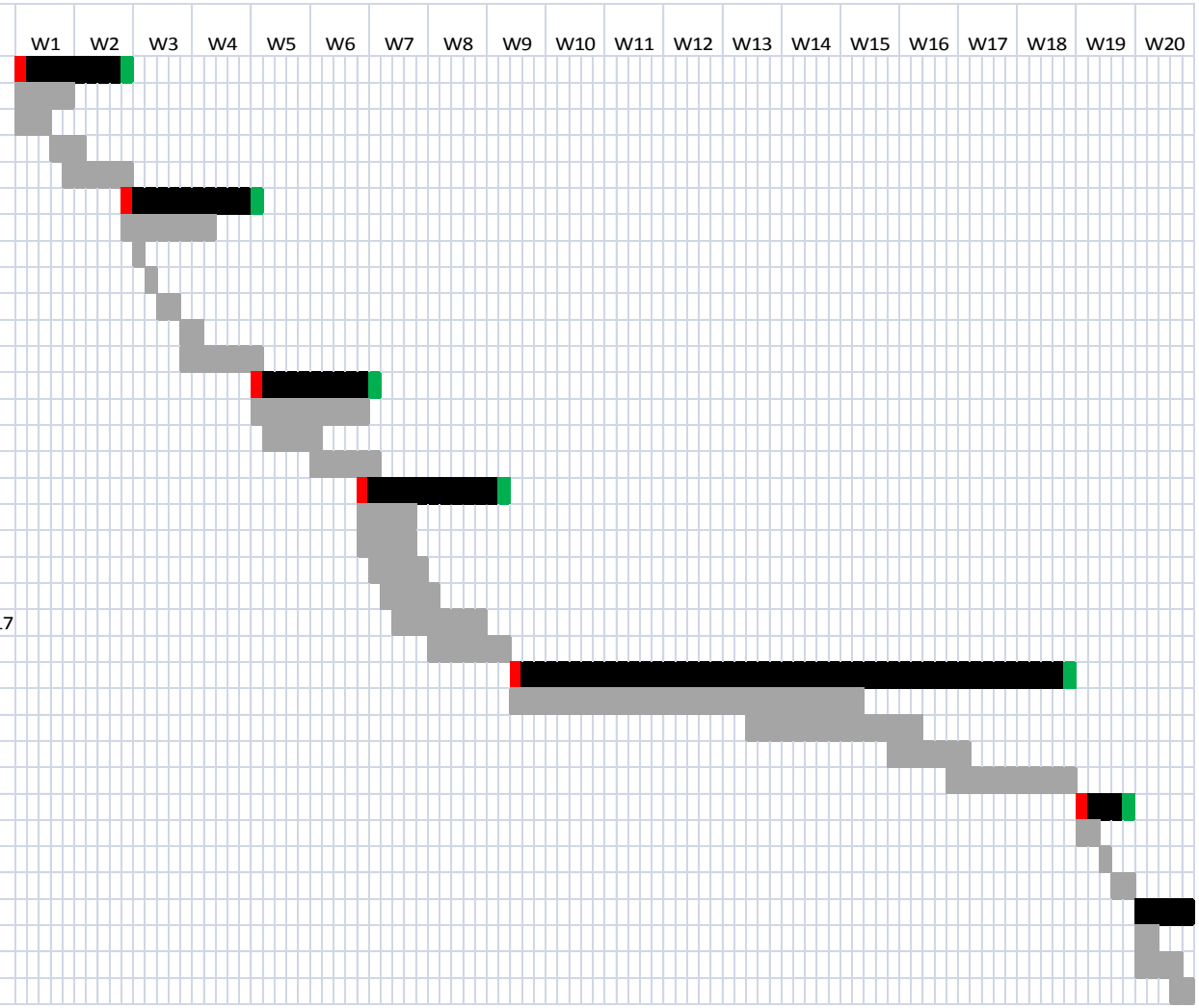



**FIGURE –31 GANTT CHART FOR ADANI SOFTWARE**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **Lessons Learned Document Title:** title of the observation or issue raised. | | | | | | | | |
| | | | | | | | | | | |
| | | **Date:**date on which it is captured | | | | **Project Name:**name of the project to which the issue belongs | | | | |
| | | **Place:**place where issue raised | | | | **Phase:**project development phase from which the issue is noticed | | | | |
| | | **Location:**the location or region | | | | **Observer:** information about the observer | | | | |
| | | | | | | **POT:** information of person of team | | | | |
| | | | | | | | | | | |
| **Observation** | **Expected Result** | **Achieved Result** | **Discussion or analysis** | **Remedial Actions** | **Action Body** | **Remedial Action Plan** | **Impact of Remedial Action Plan** | **Result** | **Conclusion** | **Reference** |
| contain the exactly what has happened? It defines the observation. | contain the discussion about the expected result or outcome which is not achieved. | contain the information about the output which is received. | contain the information about the analysis of issue raise. Why this output received and which factors are there to affect the current result to achieve. | contain the information about remedial action to be taken to solve the issue. It describes what to do solve the issue. | the information about the action body that is going to execute the remedial action | information about the remedial action plan. It should contain all the information about how to execute the plan. | information about the impact of remedial action plan. It contains the information of the effect of the remedial action plan, whether it is of positive and negative. | contain the information about the result | contain the information in which scenario the given remedial action should be implemented. | from where the remedial actions got weather it could be person, other project, forum, or any website |
| | | | | | | | | | | |
| | | | **Status:Resolved/Unresolved** | | | | | | | |

**TABLE – 8 LESSONS LEARNED TEMPLATE**


IJCAIT is indexed with

Google Scholar BETA

Search

DOAJ DIRECTORY OF OPEN ACCESS JOURNALS

ScienceCentral.com

NEWJOUR Electronic Journals & Newsletters

Peer Review:

HOME ABOUT LOG IN REGISTER SEARCH CURRENT ARCHIVES ANNOUNCEMENTS INDEXING

*Home > Vol 3, No 2 (2013)*

# IJCAIT

IJCAIT is now indexed with Google Scholar, Getcited, DOAJ, NewJour, Researchbib, Science Central.

The International Journal of Computer Applications & Information Technology (IJCAIT) is an online bi-monthly journal that publish the original and innovative research paper in the field of Computer Science and Information Technology. It is an international forum for scientists and engineers involved in all aspects of computer science and technology to publish high quality and peer reviewed papers. The objectives of IJCAIT is to provide open access Research Articles, survey and review articles from experts in the field, promoting insight and understanding of the state of the art, and trends in current technology.

USER

Username

Password

Remember me

Log In

INFORMATION

For Readers
For Authors
For Librarians

OPEN JOURNAL SYSTEMS

JOURNAL CONTENT

Search

All

INTERNATIONAL JOURNAL OF COMPUTER APPLICATIONS
IJCAIT
IJCAIT
Scholarly Peer Review Publishing System
ISSN:2278-7720



# VOL 3, NO 2 (2013)

## IJCAIT AUG-SEPTEMBER 2013

## TABLE OF CONTENTS

### ARTICLES



**IJCAIT is indexed with**

Google Scholar BETA

Search

DOAJ DIRECTORY OF OPEN ACCESS JOURNALS

ScienceCentral.com

NEW JOUR Electronic Journals & Newsletters

**Peer Review:**

IJCAIT introduces peer-review from its fist issue onwards. The researchers submitting

USER

Username

Password

Remember me

Log In

INFORMATION

For Readers
For Authors
For Librarians

OPEN JOURNAL SYSTEMS

JOURNAL CONTENT

Search

All

Search